

مركز البحوث والدراسات

تصميم وتطبيق نظم قواعد البيانات العلاقية

تأليف
أ.د. يوسف بن جاسم الهميلي



الطبعة الثانية

بسم الله الرحمن الرحيم



مركز البحوث والدراسات

تصميم وتطبيق نظم
قواعد البيانات العلاقية

تأليف

أ د يوسف بن جاسم الهميلي

الطبعة الثانية

(مزيدة ومُنقحة)

1442هـ / 2021م

بطاقة الفهرسة

ⓒ معهد الإدارة العامة، 1442هـ

فهرسة مكتبة الملك فهد الوطنية أثناء النشر

الهميلي، يوسف بن جاسم

تصميم وتطبيق نظم قواعد البيانات العلاقية / يوسف بن جاسم الهميلي - ط2 - الرياض، 1442هـ

628 ص؛ 17 × 24 سم

ردمك: 978-603-8276-30-3

1- قواعد المعلومات - تصميم 2- قواعد المعلومات - تنظيم وإدارة أ- العنوان

ديوي: 00574 1442/3290

رقم الإيداع: 1442/3290

ردمـــــك: 978-603-8276-30-3

قائمة المحتويات

مقدمة

تُعَدُّ المعلومات في وقتنا الراهن أحدَ أهمِّ موارد المنظمات؛ وذلك على اختلاف مجالات عملها ونظراً لأن المادة الأساسية التي تُستَقى وتُستَنبط منها المعلومات هي البيانات؛ فقد حدا هذا بالمنظمات المختلفة إلى جَمْع البيانات، سواء التاريخية منها أو الحديثة، وتخزينها في أماكن تضمن المحافظة عليها من العبث أو التخريب هذا بالإضافة إلى العمل الدؤوب على تحديث هذه البيانات؛ لكونها أحد الموارد الرئيسية والمهمة للمنظمات ومع تزايد أهمية البيانات في المنظمات الحديثة ظهرت الحاجة الماسة إلى مكننة طرق حِفظ واسترجاع البيانات بالإضافة إلى معالجتها؛ حتى أضحت هذه الطرق جوهر أيِّ نظام معلوماتي حديث كما أضحت النظم التي تقوم بحفظ واسترجاع ومعالجة البيانات تُعرَف بنظم إدارة قواعد البيانات وبات مجال نظم قواعد البيانات واحداً من أخصب التخصصات العلمية طَرْقاً من قِبل الباحثين، وتطبيقاً من قِبل المتخصصين في مجال تطوير النظم المعلوماتية؛ لكون هذا المجال يتعامل مع كافة أنواع المنظمات، ومنها - على سبيل المثال وليس الحصر- تلك التي تُعنَى بالتعليم، والرعاية الصحية، والمكتبات، والأعمال البنكية وأسواق المال، والتجارة الإلكترونية

ونتيجةً للأهمية الكبيرة لقواعد البيانات والنظم التي تُعنَى بإدارتها؛ فقد تمَّ تأليف عددٍ كبيرٍ من الكتب العلمية المتخصصة باللغة الإنجليزية في هذا المجال؛ موجهةً لفئات مختلفة من القراء، ومن ضِمنها الكتب الدراسية التي تُدرس في المراحل الجامعية المختلفة ويتوفَّر من بين هذه الكتب حالياً عددٌ لا بأس به من الكتب الجامعية التي تُعَدُّ مراجع دراسية متميزة للطلبة المتخصصين في مجال الحاسب الآلي؛ سواءٌ أكان ذلك للمرحلة الجامعية أم للسنوات الأولى من مرحلة الدراسات العليا، ومن ضِمنها تلك المدرجة ضِمن مراجع هذا الكتاب وقد ألَّف هذه الكتب أساتذةٌ

متخصصون في مجال قواعد البيانات ولهم باعٌ طويلٌ في البحث العلمي والتدريس في هذا المجال على المستوى الدولي كما أن هذه الكتب ظهرت في طبعات مختلفة حَظِيَت بالكثير من التنقيح والتطوير؛ حتى أصبحت بالشكل المتميز التي هي عليه الآن وعلى الرغم من أن مجال قواعد البيانات أصبح من المجالات التخصصية المعروفة التي لها كيانها الخاص ضِمن تخصصات علوم الحاسب الآلي؛ فإنه يُوجَد نقصٌ كبيرٌ في مكتبتنا العربية من الكتب العلمية المتخصصة التي تتعاطى مع مجال قواعد البيانات بشكلٍ علمي يتسمُ بالعمق والشمولية؛ من حيث العرض والتحليل والنقاش للمواضيع الرئيسية التي تدور حولها مفاهيمُ وتقنيات قواعد البيانات ونظمها وقد شجع هذا القصور النسبي على تأليف هذا الكتاب الذي يشتملُ على المواضيع الرئيسية لقواعد البيانات وبشكلٍ يميل إلى الجانب التطبيقي ودون تقصير في عرض الجوانب النظرية التي تستند إليها مفاهيم وتقنيات قواعد البيانات ولقد صُرف الوقت الكثير والجهد الكبير في تأليف هذا الكتاب؛ ليكون بادرةً تهدف إلى نقل شيءٍ مما يزخرُ به مجال قواعد البيانات من مفاهيم وتقنيات إلى المكتبة العربية، وعلى أمل أن يشكل خطوةً جادةً تساعد على إضافة المزيد من الخطوات المستقبلية إلى الأمام وصولاً إلى كتب علمية متخصصة في هذا المجال تثرى مقتنيات مكتبتنا العربية، وتسهم في إثرائها للفكر العربي المتخصص

ويتكوَّن هذا الكتاب من تسعة فصول بالإضافة إلى ملحقين يستعرض الفصل الأول الميزات التي تتحَلَّى بها نظم قواعد البيانات في بناء وتطوير نظم التطبيقات مقارنةً بالطريقة التقليدية المبنية على الملفات في تطوير النظم المعلوماتية، كما يستعرض الخطوات الرئيسية المتبعة في تحليل، وتصميم، وبناء، وإدارة قواعد البيانات ولأن قاعدة بيانات أيِّ نظام معلوماتي تمثل جزءاً من النظام المعلوماتي؛ فإن هذا الفصل يُبيِّن أيضاً موقع عملية تطوير قاعدة البيانات ضِمن عملية التطوير الكلِّي للنظام المعلوماتي ويمكن أن نلخِّص أهمَّ عمليات تطوير أيَّة قاعدة بيانات في: النمذجة المفاهيمية لبيانات المنظمة (أو النظام المعلوماتي)، التصميم المنطقي لقاعدة البيانات، التصميم المادي لقاعدة البيانات وتعريفها، وبناء قاعدة البيانات ويُقدِّم الفصل في نهايته أهمَّ الأحداث التاريخية في تطوُّر نظم قواعد البيانات وبذلك يُعَدُّ هذا الفصل مدخلاً للتعرُّف على نظم قواعد البيانات ومَيْزاتها، كما يمثل أساساً لتسلسل بقية فصول الكتاب ومدخلاً لما تحتويه من موضوعات

الفصل الثاني خُصِّص لموضوع نمذجة بيانات المنظمة التي تُعرف عادةً بما يُسمَّى «النمذجة المفاهيمية» (ConCeptual Data Modeling) فالنمذجة المفاهيمية عبارة عن نمذجة لبيانات المنظمة بشكلٍ عالي المستوى قريب من إدراك المستفيدين، غير المتخصِّصين، للبيانات التي يتعاملون معها والارتباطات فيما بينها ويشرحُ هذا الفصل المكوِّنات الأساسية لنموذج «كينونة - علاقة» (Entity-Relationship) الذي يُعَدُّ أكثر النماذج المفاهيمية العالية المستوى شيوعاً

خُصِّص الفصلُ الثالث لشرح مفاهيم ومكونات إضافية للنموذج المفاهيمي «كينونة - علاقة» والسبب وراء ذلك يرجع إلى تعقيد البيانات والعلاقات فيما بينها في بعض المنظمات الحديثة؛ مما يستدعي ضرورة إثراء نموذج «كينونة - علاقة» بمفاهيم تمكِّن من نمذجة البيانات الأكثر تعقيداً ومن هذه المفاهيم، التي يتطرَّق إليها الفصل، «الأنواع الرئيسية» و«الأنواع الفرعية» (Supertype/Subtype)، و«التعميم» (Generalization)، و«التخصيص» (SpeCialization)، و«التجميع» (Aggreation)

أما الفصلُ الرابع؛ فهو مُخصَّص لشرح المفاهيم الأساسية للنموذج العلاقي الذي يُعَدُّ أحدَ المحاور الرئيسية للكتاب فالنموذج العلاقي يُعَدُّ أنجح النماذج التمثيلية للبيانات وأكثرها استخداماً في نظم قواعد البيانات المتوفرة حالياً على المستوى التجاري، هذا بالإضافة إلى كونه الأكثر استخداماً في المنظمات الحديثة ومن أبرز أسباب نجاح وانتشار استخدام هذا النموذج سهولته في تمثيل البيانات، إضافةً إلى استناده إلى أسس رياضية صلبة تمكنه من التعامل مع البيانات وحساب نتائجها لذلك؛ فإن هذا الفصل يستعرض أيضاً لغتين من لغات النموذج العلاقي الرسمية، وهما: الجبر العلاقي (Relational Algebra) – التي تُعَدُّ إحدى لغات النموذج العلاقي الإجرائية (ProCedural Language)؛ والحساب العلاقي (Relational CalCulus) – التي تُعَدُّ إحدى لغات النموذج العلاقي غير الإجرائية (NonproCedural Language)

بعد التعرُّف على النمذجة المفاهيمية للبيانات (في الفصل الثاني والثالث) وعلى النموذج العلاقي ولغاته الرسمية (في الفصل الرابع)؛ يشرح الفصل الخامس مرحلة التصميم المنطقي لقواعد البيانات تعتمدُ هذه المرحلة على النموذج التمثيلي (Representational Model) المستخدَم، وهو النموذج العلاقي في هذا الكتاب وتتكوَّن مرحلة التصميم المنطقي من خطوتين رئيسيتين: في الخطوة الأولى يتمُّ تحويل النموذج المفاهيمي إلى نموذج قاعدة البيانات المستخدمة ولأن هذا

الكتاب يركِّز على قواعد البـيانات العلاقية؛ فإن هذه الخطوة تعني تحويل النموذج المفاهيمي إلى النموذج العلاقي أمَّا في الخطوة الثانية؛ فيتم تحسين تصميم قاعدة الـبيانات الناتجة من عملية التحويل؛ بحيث تحتوي على أقل قدرٍ ممكنٍ من البيانات المتكررة حتى يتم تجنُّب المشكلات التي قد تنتج عن عمليات التعديل على محتويات قاعدة البـيانات وتُدعَى هذه الخطوة بعملية «التطبيع» (Normalization) ويركِّز الفصل الخامس على الخطوة الأولى من مرحلة التصميم المنطقي

يُوضِّح الفصلُ السادس، في جزئه الأول، مفهومَ «الجداول جيدة البناء» (Well-StruCtured Relations) بـشكلٍ غير رسمي وبناءً على مفهوم «الجداول جيدة البناء» يتمُّ شرح مفهوم «التطبيع» (Normalization) الذي يمثل «طريقة رسمية» (Formal Method) واضحة المعالم لها أسسها النظرية التي تمكِّن من التعرُّف على جودة الجداول التي تمَّ تصميمها في الخطوة الأولى من التصميم المنطقي لقاعدة البـيانات ثم يشرحُ هذا الجزء من الفصل السادس مستويات التطبيع الأكثر استخداماً في تصميم قواعد البيانات؛ وذلك ابتداءً من الشكل الطبيعي الأول وانتهاءً بالشكل الطبيعي الرابع

يُركِّز الجزء الثاني من الفصل السادس على مرحلة التصميم المادي لنظم قواعد البيانات وتهدف هذه المرحلة من تصميم قواعد البيانات إلى إنشاء تصميمٍ يُمكِّن من تخزين البيانات بشكلٍ يوفِّر الأداء المناسب لنظام إدارة قاعدة البـيانات على اختلاف حجم العمليات التي تُنفَّذ عليها ويعني هذا، وعلى خِلاف التصميم المفاهيمي والتصميم المنطقي، أن التصميم المادي يوضِّح الكيفية التي ستُخزَّن وتُعالَج فيها الـبيانات لا على الكيفية التي يتمُّ من خلالها التعرُّف على البيانات والعلاقات فيما بينها أو طريقة تمثيلها وفق النموذج العلاقي أو نماذج البيانات الأخرى

خُصِّص كلا الفصلين السابع والثامن لشرح بعض مكونات «لغة الاستفسار البنائية» (StruCtured Query Language (SQL)) التي تُعَدُّ واحدةً من أكثر لغات قواعد البيانات العلاقية انتشاراً؛ بحيث تمَّ تبنِّيها من قِبل «معهد المقاييس الوطني الأمريكي» (AmeriCan National Standards Institute) و«منظمة المقاييس الدولية» (International Standards Organization (ISO)) ويمكن تقسيم تعليمات لغة الاستفسار الـبنائية إلى ثلاث مجموعات من التعليمات (أو «اللغات الفرعية» (Three Sub-Languages)) وهي: مجموعة (أو لغة) تعريف البيانات، ومجموعة (أو لغة) معالجة البيانات، ومجموعة (أو لغة) التحكم في البـيانات ونظراً

لأهمية لغة الاستفسار البنائية في قواعد البيانات العلاقية؛ فإن هذا يستلزمُ شرحَ مكوناتها الأساسية بشكلٍ مستفيضٍ يميلُ إلى الجانب التطبيقي حتى يتسنَّى فهم طريقة عمل تعليماتها

يحتوي الفصلُ السابع على شرحٍ لمجموعة تعليمات (أو لغة) تعريف البيانات وعلى شرحٍ للعبارات الأساسية في تعليمة الاختيار (أو الاسترجاع) التي تُعَدُّ من أهمِّ تعليمات لغة معالجة البيانات ويُستخدَم في هذا الفصل، والفصل الثامن كذلك، نظام إدارة قاعدة بيانات أوراكل في تنفيذ واختبار تعليمات لغة الاستفسار البنائية القياسية؛ وذلك لكون هذه البيئة واحدةً من الأوسع انتشاراً بين المتخصِّصين في تطوير التطبيقات المبنية على نظم قواعد البيانات، هذا إضافةً إلى تشابهها مع ما توفره نظم إدارة قواعد البيانات المعروفة الأخرى على المستوى التجاري من بيئات مشابهة لتنفيذ تعليمات لغة الاستفسار البنائية أمَّا بالنسبة للقراء الذين لا تتوفر لديهم بيئة أوراكل، وحتى يتمكن هؤلاء أيضاً من تطبيق مفاهيم وتعليمات لغة الاستفسار البنائية والاستفادة القصوى من محتويات هذا الفصل والفصل الثامن، فيُستخدَم في هذا الفصل نظامُ إدارة قاعدة بيانات «أكسس» (ACCESS)، بشكلٍ مُقتَضب؛ وذلك لإعطاء فكرة مُبسَّطة لطريقة التعامل مع هذا النظام، ولتنفيذ تعليمات لغة الاستفسار البنائية والسبب الوحيدُ وراء استخدام نظام إدارة قاعدة بيانات أكسس هو توفُّره في غالبية الحاسبات الشخصية في وقتنا الراهن، ومن ثم يمكِّن القارئ من الاستفادة القصوى من محتويات الفصول المتعلقة بلغة الاستفسار البنائية من خلال التطبيق العملي

يستكملُ الفصلُ الثامن شرحَ مكوِّنات لغة الاستفسار البنائية؛ إذ خُصِّص الجزءُ الأول منه لاستكمال شرح تعليمة الاختيار (أو الاسترجاع)، عندما تقوم التعليمة بالتعامل مع أكثر من جدولٍ في آنٍ واحد، ولشرح بقية مجموعة تعليمات (أو لغة) معالجة البيانات أمَّا الجزء الثاني من الفصل؛ فقد خُصِّص لشرح مجموعة تعليمات (أو لغة) التحكم في البيانات

يتطرَّق الفصل التاسع، وبشكلٍ مُقتضب، إلى أربعةِ موضوعات متطوِّرة في نظم قواعد البيانات، هي: المعاملات، وقواعد البيانات الشيئية، وقواعد البيانات العلاقية - الشيئية، وقواعد البيانات الموزَّعة تمثل المعاملات الوسيلة الرئيسية التي يتمُّ من خلالها التفاعل مع قواعد البيانات من قِبل المستفيدين، سواء بشكلٍ تفاعلي مباشر أو من خلال برامج التطبيقات التي يقوم مطورو التطبيقات ببنائها أمَّا نموذج البيانات الشيئي؛ فقد تمَّ تطويره لسد الاحتياجات التقنية التي يتطلبها تطوير نظم التطبيقات المتعلقة بمكننة أعمال المنظمات ذات الصبغة غير التقليدية من حيث

البيانات التي تتعامل معها، مثل: استخدامها لتطبيقات «التصميم بمساعدة الحاسب الآلي» (Computer-Aided Design)، و «التصنيع بمساعدة الحاسب الآلي» (Computer-Aided ManufaCturing)، و«نظم المعلومات الجغرافية» (GeographiCal Information Systems)، و«تطبيقات الوسائط المتعددة» (Multimedia AppliCations)؛ على سبيل المثال فحسب ونظراً لانتشار النموذج العلاقي وسهولته في تمثيل البيانات والتعامل معها؛ فقد عَكَفَ الكثيرُ من الشركات المصنِّعة لنظم إدارة قواعد البيانات العلاقية على تبنِّي بعض مفاهيم النموذج الشيئي ضِمن منتجاتها؛ حتى تتمكَّن من مواكبة احتياجات المنظمات التي تتصف بياناتها بالصِّبغة غير التقليدية إضافةً إلى تلك التي تتصف بالتقليدية وأصبحت مثل هذه المنتجات تُسمَّى قواعد البيانات العلاقية ـ الشيئية أمَّا بالنسبة للمنظمات التي تتوزع مقارُّها في مناطق عديدة، وعلى رُقع متباعدة جغرافياً في الكثير من الأحيان؛ فقد دفعت هذه المنظمات الباحثين إلى تبنِّي مفهوم النظم الموزَّعة وأضحت تُسمَّى في مجال نظم قواعد البيانات «نظم قواعد البيانات الموزَّعة» (Distributed Database Systems) وتوفر مثل هذه النظم العديد من الميزات مقارنةً بتلك النظم المركزية من ضِمنها «الموثوقية» (Reliability) و«التواجد» (Availability) هذا بالإضافة إلى أدائها المتميِّز وسهولة التوسُّع في الأجهزة والتطبيقات في مثل هذه المنظمات ونظراً لأهمية المفاهيم السابقة كان من الضروري التطرُّق إليها في هذا الكتاب ولو بشكلٍ مُقتضَب

يحتوي الملحق رقم (1) على حالةٍ دراسية تمثل نظاماً مفترضاً للتسجيل في إحدى الجامعات الأهلية ويلخِّص الملحق عملية النمذجة المفاهيمية، والتصميم المنطقي، والتصميم المادي لقاعدة بيانات النظام وبذلك؛ فإن هذا الملحق يمثل تطبيقاً للعديد من المفاهيم الواردة في الكتاب في محاولة للاستفادة القصوى منه كما يحتوي هذا الملحق على تمارين تطبيقية على لغة الاستفسار البنائية؛ كونها من أهمِّ مكونات نظم قواعد البيانات العلاقية وتتفاوت صعوبة هذه التمارين من البسيطة جداً وحتى الصعبة جداً وتهدف هذه التمارين في مجملها إلى اختبار قدرة القارئ واستيعابه لمكونات لغة الاستفسار البنائية، من جهة، وإلى زيادةٍ تمكِّنه من اللغة، من جهة أخرى

خُصِّص الملحق رقم (2) لحالة دراسية أخرى تتمثل في نظام افتراضي لإحدى شركات بيع وتأجير العقارات ويهدف هذا الملحق إلى تعزيز قدرة القارئ من النمذجة المفاهيمية، والتصميم المنطقي، والتصميم المادي كما يُغطي هذا الملحق بعض المفاهيم التي لم يغطها الملحق

رقم (1) وإضافةً لذلك؛ يوفر هذا الملحق بعض التمارين التطبيقية على لغة الاستفسار البنائية وقد تمَّ إدراج كلتا قاعدتي البيانات المستخدمتَيْن في الملحق رقم (1) والملحق رقم (2) في الرابط والباركود المُشَار إليهما أدناه

وبناءً على الاستعراض السريع لمحتويات هذا الكتاب؛ يتضحُ أنه قد خُصِّص، وبشكلٍ رئيسيٍّ، للطلبة الدارسين في مواد نظم قواعد البيانات من المتخصِّصين في مجال الحاسب الآلي؛ سواءٌ في مرحلة الدراسة الجامعية (البكالوريوس) أو طلبة الدبلوم (فوق الثانوي) كما أنه يستهدفُ أيضاً مُطوِّري نظم التطبيقات الذين يتعاملون مع نظم قواعد البيانات العلاقية بشكلٍ تطبيقي في حياتهم اليومية؛ ليكون مرجعاً تطبيقياً لهم وقد رُوعِي في الكتاب استخدامُ المصطلحات الإنجليزية مع ما يقابلها في العربية؛ حتى يتمكَّن القارئ من فهم هذه المصطلحات في حالِ عدم دقة الترجمة للمصطلح الإنجليزي، وحتى يتمكَّن من الرَّبط بينهما عند الرجوع إلى أيِّ مطبوعات أخرى مكتوبة باللغة الإنجليزية كما تمَّ إدراجُ قائمةٍ بأهمِّ المصطلحات الإنجليزية الواردة في هذا الكتاب وما يقابلها باللغة العربية في الملحق رقم (3)

ويأملُ المؤلف أن يلاقي هذا الكتاب قبولاً واستحساناً من الفئة المُستَهدَفة من القراء، كما يتطلع إلى مقترحاتهم وآرائهم وانتقاداتهم البنَّاءة التي من شأنها أن تُثري محتويات هذا الكتاب في أية طبعة مستقبلية، سائلاً المولى عز وجل أن ينفع به ويفيد منه إنه السميع العليم

المؤلف

أ د يوسف بن جاسم الهميلي

houmaily@ipaedusa

https://wwwipaedusa/ar-sa/knowledge/DoCuments/BookAppendixzip



رابط قاعدتَيْ البيانات المُحمَّل بالباركود

امسح الباركود للحصول على رابط قاعدتَيْ البيانات

المُستخدَمتَيْن في الملحق رقم (1) والملحق رقم (2).

الفصل الأول

تطوير نظم المعلومات

صُمِّمت الحاسباتُ الآلية عند بداية ظهورها لحساب الدوال الرياضية المُستخدَمة في التطبيقات العلمية والعسكرية التي تستنزف من الوقت الكثير لحسابها يدوياً. وكان ذلك بعد انتهاء الحرب العالمية الثانية. ولقد بدا واضحاً منذ بداية ظهور الحاسبات الآلية أهميتها في حِفظ ومعالجة البيانات؛ إلا أنها لم تُستخدَم في التطبيقات الإدارية للمنظمات المختلفة حتى بداية الخمسينيات الميلادية. ومن أسباب ذلك غلاء أسعار الحاسبات الآلية في تلك الفترة وندرة المتخصِّصين في التعامل معها. لذا؛ فقد انحصر استخدام الحاسبات الآلية على فئات محدودة من المختصين الذين تتطلب طبيعة أعمالهم قدراتٍ حسابيةً عالية. وكانت التطبيقات في ذلك الوقت تتصف بمعالجتها لبيانات كثيرة وبشكلٍ متكرر، ولذلك كان استخدام الحاسب الآلي مبرراً في مثل هذه التطبيقات. ومع التطور المستمر والحثيث للحاسبات الآلية بالإضافة إلى التقلص المستمر في أسعارها وزيادة أعداد المتخصصين في التعامل معها؛ أصبحت الحاسبات تُستَخدَم بشكلٍ مكثف في غالبية المنظمات الحديثة لما تقدِّمه من مَيْزات جمَّة مقارنةً بالنظم اليدوية.

وأُستخدِمت في بداية ظهور الحاسبات الآلية الملفاتُ لتصبح وسيلةً رئيسيةً لتخزين البيانات. وكان كلُّ نظام تطبيقي يتعامل مع الملفات الخاصة به فقط. غير أن الزيادة المطردة في أحجام البيانات وضرورة المشاركة في استخدامها من قِبل تطبيقات مختلفة؛ وذلك نتيجة طبيعية لزيادة أعداد المستفيدين من الحاسبات الآلية وتطوُّر نظم التطبيقات - أدى إلى ظهور ما يُعرَف اليوم بنظم قواعد البيانات. وتتميَّز النظم المبنية على قواعد البيانات عن النظم المبنية على الملفات بعددٍ

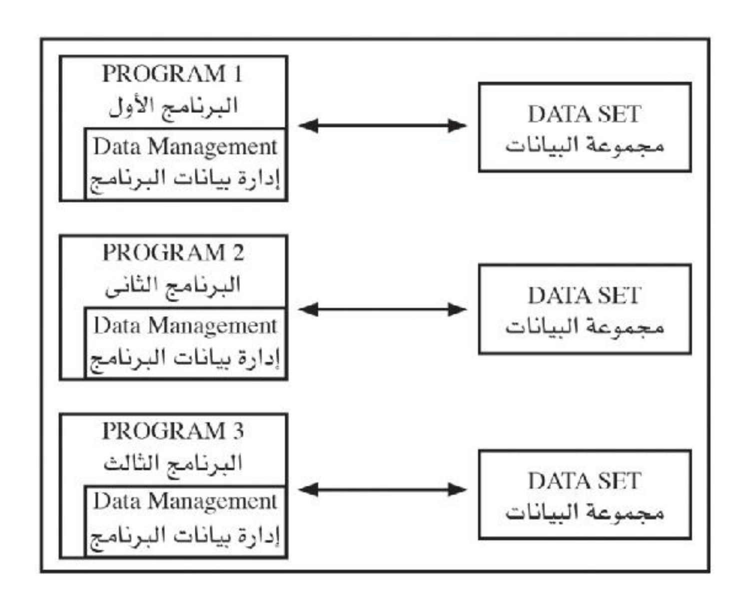
من المزايا الجوهرية التي أدَّت إلى تطوُّر نظم قواعد البيانات؛ بحيث أضحى هذا المجال واحداً من أكثر المجالات العلمية بحثاً وتطبيقاً في وقتنا الراهن.

ويستعرض هذا الفصل الميزات التي تتحلى بها نظم قواعد البيانات في بناء وتطوير نظم التطبيقات مقارنةً بالطريقة التقليدية لتطوير النظم المعلوماتية. كما يستعرض هذا الفصل الخطوات الرئيسية المتبعة في تحليل، وتصميم، وبناء، وإدارة قواعد البيانات. ولأن قاعدة بيانات أيِّ نظام معلوماتي تمثل جزءاً من النظام المعلوماتي؛ فإن هذا الفصل يُبيّن موقع عملية تطوير قاعدة البيانات ضِمن عملية التطوير الكلي للنظام المعلوماتي. ويلخص هذا الفصل في نهايته أهمَّ الأحداث التاريخية في تطور نظم قواعد البيانات.

1-1 مَيْزات النظم التطبيقية المبنية على قواعد البيانات:

1-1-1 تطوير النظم باستخدام الملفات التقليدية:

عند بداية ظهور الحاسبات الآلية في معالجة البيانات لم يكن هناك ما يُعرَف بقاعدة البيانات، وإنما كانت مجموعة البيانات المتعلقة ببرنامج حاسوبي مُعيَّن تدخل إلى الذاكرة الرئيسية للحاسب الآلي بشكلٍ مباشر، وفي الوقت نفسه الذي تدخل فيه تعليمات البرنامج الذي سيقوم بمعالجتها، كما هو مُوضَّح بالشكل رقم (1–1).



شكل رقم (1–1): ارتباط البيانات بالبرامج عند بداية ظهور الحاسبات الآلية.

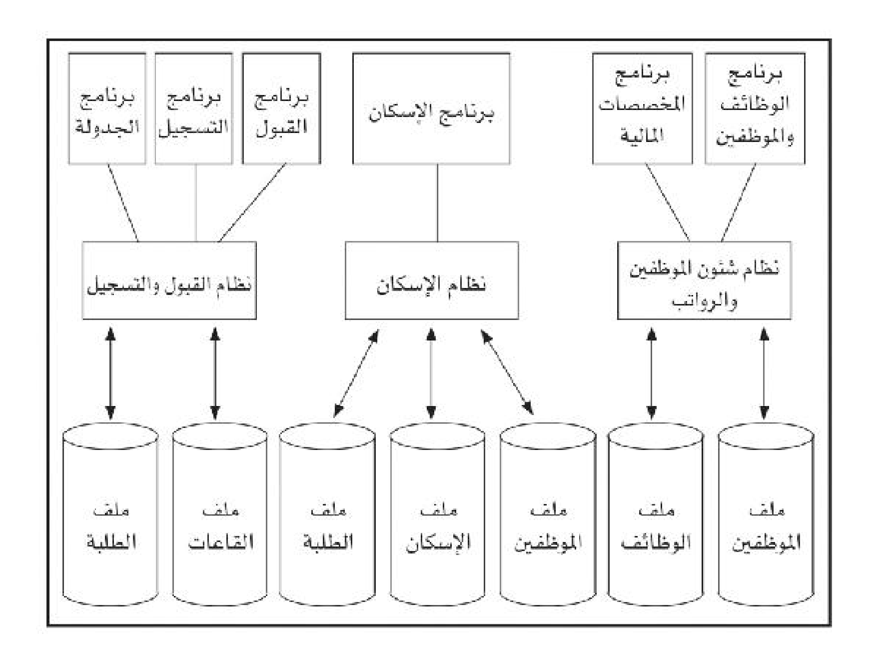
وعند انتهاء تنفيذ البرنامج تُطبع نتائج معالجة البيانات، التي قام بها البرنامج، ويتمُّ مَسْح البرنامج مع بياناته من الذاكرة الرئيسية للحاسب الآلي. ويعني هذا أنه لم يكن موجوداً في تلك الفترة ما يُعرَف بالقرص الصلب، أو الذاكرة الثانوية، الذي يحتوي على بيانات البرنامج بشكلٍ دائم، وإنما كانت تدخل البرامج مع بياناتها في كلِّ مرة يتمُّ فيها تنفيذ البرنامج. ومع تطوُّر تطبيقات الحاسب الآلي وكبر حجم البرامج التي تتطلبها هذه التطبيقات إضافةً إلى كبر حجم البيانات التي تقوم بمعالجتها؛ أصبح من المتعذر إدخالها في الذاكرة الرئيسية للحاسب الآلي. إضافةً إلى ذلك؛ فإنه لم يكن معروفاً في تلك الفترة ما يُعرَف بمبدأ المشاركة بين المستفيدين للحاسب الآلي، وإنما كان الحاسب الآلي مُسخراً لخدمة برنامج واحد في كلِّ مرة. وعند انتهاء برنامج ما يدخل برنامج آخر مع بياناته، وهكذا.

ونتيجةً لهذا القصور ظهرت الذاكرة الثانوية بشكلٍ عام، والقرص الصلب على وجه الخصوص. كما ظهرت نظم التشغيل التي تسمح بمبدأ المشاركة. وسمح هذا التطور بتخزين البيانات في ملفات على القرص الصلب للحاسب الآلي. ومع هذا التطور بدأ الحاسب الآلي بالانتشار بشكلٍ واسعٍ حتى في التطبيقات الإدارية والمالية لمعظم المنظمات. وباستخدام هذا النمط الحاسوبي؛ كانت التطبيقات تُجزَّأ إلى مجموعة من البرامج؛ بحيث يقوم كلُّ برنامج بوظيفةٍ واحدةٍ أو مجموعةٍ من الوظائف المترابطة منطقياً فيما بينها، كما كان لكلِّ تطبيقٍ مجموعته الخاصة من الملفات التي تحتفظ ببياناته على القرص الصلب. ويمثل الشكل رقم (1–2) ثلاثة تطبيقات إدارية ومالية لإحدى المنظمات التعليمية الافتراضية. يُعنى النظام الأول، وهو نظام القبول والتسجيل، بقبول الدارسين وتسجيلهم في المواد الدراسية المختلفة إضافةً إلى جدولة القاعات الدراسية المتوفرة في المنظمة. وتمَّ تقسيمُ النظام إلى ثلاثة برامج فرعية، كلٌّ واحدٍ منها يُعنَى بمجموعةٍ من الوظائف المترابطة منطقياً فيما بينها. كما تمَّ تعريف ملفين رئيسيين للنظام: أحدهما لتخزين البيانات الخاصة بالطلبة، والثاني لتخزين البيانات المتعلقة بالقاعات الدراسية.

أمَّا النظام الثاني؛ فقد خُصِّص لإسكان أعضاء هيئة التدريس وإسكان الدارسين، ويتكون من برنامج واحد. كما أن هذا النظام مرتبطٌ بثلاثة ملفات رئيسية: أحدها لتخزين البيانات الخاصة بالطلبة، والثاني لتخزين البيانات المتعلقة بالمساكن المتوفرة وبيانات ساكنيها، والثالث لتخزين البيانات المتعلقة بالموظفين.

النظام الثالث في هذه المنظمة؛ هو نظام شئون الموظفين والرواتب الذي يتكون من برنامجين فرعيين: أحدهما يُعنَى بالوظائف المتوفرة في المنظمة والموظفين المعيَّنين عليها، أما

الثاني فيُعنَى بالمخصصات المالية المختلفة للموظفين. ويرتبط هذا النظام بملفين: أحدهما مخصصٌ للوظائف المتوفرة، والثاني مخصصٌ لبيانات الموظفين.



شكل رقم (1–2): تخزين البيانات في ملفات على القرص الصلب واستخدام مبدأ المشاركة.

ولا شك أن النُّظم السابقة نظمٌ جيدةٌ تؤدي الأهداف التي صُمِّمت من أجلها؛ غير أن هذه النظم تعاني من مشكلات جوهرية؛ لأنها ترتكزُ في تصميمها وبنائها على الملفات التقليدية. ويمكن تلخيص هذه المشكلات فيما يلي:

- الاعتمادية (أو الترابط) بين البرامج والبيانات (Program-Data DependenCe):

باستخدام الملفات التقليدية يجب توصيف الملفات المُستخدَمة لتخزين البيانات داخل برامج النظام التطبيقي؛ فعلى سبيل المثال: يجب وَصْف ملف القاعات الدراسية، وملف الطلبة في برنامج الجدولة، كما يجب وَصْف ملف الطلبة في كلٍّ من برنامج القبول وبرنامج التسجيل. ويُقصَد بوَصْف البرنامج تحديدُ طبيعة الملف إذا كان «متسلسلاً» (Sequential File)، أو ذا «استرجاع عشوائي» (Random ACCess File). كذلك يُقصَد بوَصْف الملف تعريف البيانات التي ستُخزَّن عليه من حيث مسمياتها ونوعية بياناتها (أعداد «صحيحة» (Integers)، أعداد «حقيقية» (Real (or Float)، «سلاسل حرفية» (CharaCter String)... إلخ).

ويُقصَد بالاعتمادية بين البرامج والبيانات أنَّ كلَّ برنامج يجب أن يحتوي على توصيفٍ للملفات التي يتعاملُ معها من حيث نوعيتها، وتركيبها، وطريقة الوصول إلى البيانات المخزَّنة عليها. وعند وجود ما يستدعي تغيير مكوِّنات أحد الملفات؛ فإن هذا يتطلَّب إعادة توصيف الملف قَيْد التغيير في كافة البرامج التي تتعامل مع الملف. ولنفترض على سبيل المثال أنه وَجَب تغييرُ طول حقل مُسمَّى اسم العائلة للطلبة من عشرة حروف إلى خمسة عشر حرفاً؛ نتيجةً لقبول أحد الطلبة الذي يشكِّل اسم عائلته أكثر من عشرة حروف. في هذه الحالة يجب تغيير سجل توصيف ملف الطلبة ليس في برنامج القبول فحسب، وإنما في برنامج التسجيل وبرنامج الجدولة أيضاً؛ لأن كلاً من هذه البرامج يتعامل مع ملف الطلبة.

- تكرارية البيانات (Data DupliCation):

يتمُّ تطويرُ النظم باستخدام الملفات التقليدية عادةً كلٌّ على حِدة؛ مما يعني أنَّ القاعدة الرئيسية في مثل هذه النظم هي تكرارية البيانات. فعلى سبيل المثال يحتوي ملف الإسكان على بيانات بعض الطلبة وبيانات بعض أعضاء هيئة التدريس، وهم الذين يمثلون مجموعة السكان من ضِمن المجموعة الكلية للطلبة وأعضاء هيئة التدريس. وينجم عن هذه التكرارية، في غالبية الأحيان، ضياعٌ كبيرٌ للطاقة التخزينية للحاسبات. ليس هذا فحسب، وإنما ينجم عن هذا ضياعٌ «لتكامل البيانات» (Data Integrity) التي يُقصَد بها إما عدم توافق «هيئة البيانات» (Data Format) المخزَّنة أو عدم توافق القيم المخزَّنة للبيانات نفسها أو الاثنين معاً. فعلى سبيل المثال لنفترض أنَّ رقم الهاتف مُعرَّفٌ على أنه «سلسلة حرفية» (CharaCter String) مكوَّنة من سبعة حروف في ملف الطلبة التابع لنظام القبول والتسجيل وملف الموظفين التابع لنظام شئون الموظفين والرواتب، وأن هذا الحقل نفسه مُعرَّف على أنه عددٌ «صحيح» (Integer) في كلٍّ من ملف الطلبة وملف الموظفين التابعين لنظام الإسكان. يمثل عدم التوافق هذا في تعريف الحقل نفسه ضياعاً لتكامل البيانات؛ إذ إن هيئة الحقل يجب أن تكون واحدةً بغض النظر عن البرنامج الذي يتعامل معه. ولإيضاح الجانب الثاني لضياع تكامل البيانات لنفترض انتقال أحد الطلبة من مسكنٍ إلى آخر الأمر الذي قد يتطلب تغيير رقم هاتف الطالب وعنوانه البريدي. ولأن ملف الطلبة يتبع لنظامٍ آخر وهو نظام القبول والتسجيل؛ فإن تغيير رقم هاتف الطالب وعنوانه البريدي قد يتمُّ ضِمن نظام الإسكان؛

ولكن القيم القديمة لرقم هاتف الطالب وعنوانه البريدي ستبقى كما هي ضِمن ملف الطلبة التابع لنظام القبول والتسجيل. ويعني هذا ضياعاً لتكامل البيانات من حيث القيم المخزَّنة فيها.

- المحدودية في مشاركة البيانات (Limited Data Sharing):

لكلِّ نظام ملفاته الخاصة به عند استخدام طريقة تطوير النظم بالملفات التقليدية. ويعني هذا أن المستفيدين من نظامٍ معين؛ ليس بإمكانهم التعامل مع البيانات المخزَّنة ضِمن ملفات التطبيقات الأخرى بشكلٍ مباشر؛ الأمر الذي تتطلبه طبيعة العمل في الغالبية العظمى من الأحيان. فعلى سبيل المثال؛ لنفترض أنه قد وَجَب إجراءُ خصمٍ معين من مُخصَّصات أعضاء هيئة التدريس كافة الذين تتوفر لهم مساكن من خلال إدارة الإسكان بالمنظمة. في هذه الحالة؛ يجب إجراء تعديلات على برنامج المخصَّصات المالية التابع لنظام شئون الموظفين والرواتب. ويستلزم مثل هذا الإجراء توصيف ملف الإسكان ضِمن برنامج المخصَّصات المالية للموظفين، الذي يُعَدُّ إجراءً عادياً في مثل هذه البيئة الحاسوبية. ولكن المشكلة الحقيقية التي تظهرُ بوضوح في مثل هذه الحالة، هي عدم توافق ملف الإسكان مع بقية ملفات نظام شئون الموظفين والرواتب، مثل: تعريف طول اسم الموظف في نظام الإسكان مع تعريف طول اسم الموظف في نظام شئون الموظفين والرواتب. وتَحُدُّ مثل هذه المشكلة من مشاركة البيانات بين التطبيقات المختلفة؛ بسبب طول الفترة الزمنية اللازمة لكتابة برامج جديدة أو تعديل ما هو متوفر منها للتغلُّب على مشكلة عدم التوافق. وحتى لو سخر الوقت اللازم لكتابة البرامج اللازمة أو إجراء التعديلات على ما هو متوفر منها؛ فإن مثل هذا الإجراء قد يُضحِّي بسرِّية البيانات التي قد تكون هاجساً كبيراً في بعض المنظمات. فعلى سبيل المثال: لو عُدِّل نظام الإسكان عوضاً عن نظام شئون الموظفين والرواتب للقيام بالخصم اللازم من أعضاء هيئة التدريس؛ فإن مثل هذا الإجراء قد يُضحِّي بسرِّية المخصصات المالية للموظفين؛ بحيث إن العاملين بإدارة الإسكان قد يكون بإمكانهم الاطلاع على مثل هذه البيانات.

ونتيجةً لما سبق؛ فإن مبدأ مشاركة البيانات بين التطبيقات المختلفة يكون محدوداً جداً في النظم التي تعتمد في بنائها على الملفات التقليدية.

– التطويل في عملية التطوير (Lengthy Development Time):

لكون كلِّ تطبيقٍ له ملفاته الخاصة التي تحتوي على بياناته؛ فإنه تُوجَد إمكانيةٌ محدودةٌ للاستفادة من المجهودات السابقة التي بذلت في تصميم نظمٍ أخرى؛ إذ إن تطوير التطبيقات يبدأ عادةً من نقطة الصفر؛ بحيث يجب تصميم ملفات النظام الجديد متضمناً ذلك «هيئتها» (Format) وتوصيفها، وبرمجة الطرق التي من خلالها سيتمُّ الوصول إلى البيانات. ولأن عملية تطوير تطبيقات جديدة لا تستفيد كثيراً من المجهودات السابقة؛ فإن هذا يؤدي إلى الإطالة في عمليات التطوير؛ حتى تصبح النظم الجديدة قيد الاستخدام التشغيلي الفعلي.

- تكثيف صيانة البرامج (ExCessive Program MaintenanCe):

إن المساوئ السابقة مجتمعة تؤدي إلى تكثيفٍ في صيانة برامج التطبيقات. ومن أكثر أسباب تكثيف صيانة البرامج في مثل هذه البيئة؛ هو كون كلِّ نظامٍ مسئولاً عن عملية تهيئة بياناته وتخزينها وصيانتها بمعنى أنَّ كلَّ نظام مسئول عن تعريف الملفات الخاصة به وهيئة البيانات المخزَّنة في كلِّ ملف. وعند إجراء أيِّ تعديلٍ على تعريف السجلات، مثل إضافة أو حذف حقل ما أو حتى تغيير طول الحقل أو نوعية بياناته؛ فإن هذا يستدعي صيانة البرامج المسئولة عن قراءة البيانات من الملف والبرامج المسئولة عن كتابة البيانات عليه. ولأن عملية صيانة التطبيقات بشكلٍ مكثفٍ تستنزف مجهودات المسئولين عن هذه النظم من الفنيين؛ فإن هذا يَحِدُّ بشكلٍ كبيرٍ من الشروع في تطوير تطبيقات جديدة تخدم أهداف المنظمة.

2-1-1 تطوير النظم باستخدام قواعد البيانات:

يوفر استخدامُ نظم إدارة قواعد البيانات في تطوير التطبيقات عدداً من الميزات على الطريقة التقليدية في تطوير التطبيقات. ويمكن تلخيص هذه الميزات فيما يلي:

- استقلالية البيانات عن برامج التطبيقات (Program-Data IndependenCe):

يتمُّ وصف البيانات المخزَّنة فعلياً في قاعدة البيانات بما يُعرَف «بالبيانات الوصفية» (Metadata). وتُخزَّن البيانات الوصفية وأشياء أخرى تخصُّ نظام إدارة قاعدة البيانات، بما يُعرَف «بالمستودع» (Repository). ويتمُّ تطويرُ التطبيقات بمعزل عن البيانات الوصفية التي تصف البيانات الفعلية المخزَّنة في قاعدة البيانات. لذلك؛ فإن هنالك استقلاليةً بين برامج التطبيقات

والبيانات المخزَّنة في قاعدة البيانات، بمعنى أن نظم التطبيقات معزولة عن طريقة تمثيل البيانات وهياكلها في قاعدة البـيانات، وكذلك طريقة تخزينها. لذا؛ فإن نظم إدارة قواعد الـبيانات تسمح بتطوُّر قاعدة البيانات ونموِّها دون أيِّ إخلال أو تأثير في التطبـيقات التي تم تطويرها.

- التحكم في تكرارية البيانات (Controlled DupliCation of Data):

تهدف نظم إدارة قواعد البيانات إلى تجميع البيانات المخزَّنة في ملفات منفصلة وإزالة أية ازدواجية فيها من خلال تمثيلها في هيكلٍ منطقيٍّ واحد؛ حيث يتمُّ تسجيل بـيانات كل حقيقة في مكانٍ واحد فقط. وعندما يلزم تكرارُ بعض البيانات الممثلة لحقائق معينة؛ فإن مثل هذا التكرار يكون محدوداً ومحكوماً من قِبل نظام إدارة قاعدة البـيانات.

- تحسين تناسق البيانات (Improved Data ConsistenCy):

تُحسِّن نظم إدارة قواعد البيانات من مستوى تناسق البيانات؛ إذ إنها تلغي ازدواجية البيانات أو تتحكم في الازدواجية عند وجودها. فعند تغيير أحد الموظفين أو الطلـبة لعنوانه البريدي، على سبيل المثال، فإن هذا التغيير يتم من خلال تعديل بيانات الموظف أو الطالب في مكانٍ واحد فقط، ومن ثم؛ فإن مثل هذا التعديل لا يؤدي إلى عدم تناسق للبيانات؛ مما قد يحدث عند استخدام الملفات التقليدية التي تتكرر فيها البـيانات وتؤدي إلى عدم تناسقها. كما أن عملية التحديث تكون أيسر مقارنةً بالملفات التقليدية؛ لكونها تتم في مكانٍ واحد، ناهيك عن التغلُّب على المساحة التخزينية الضائعة نتيجةً لتكرارية البـيانات في الملفات التقليدية.

- تحسين مشاركة البيانات (Improved Data Sharing):

إنَّ جميع بيانات المنظمة تُخزَّن في مكانٍ واحدٍ عند استخدام نظم إدارة قواعد البيانات عوضاً عن تخزين البيانات الخاصة بكلِّ تطبيق في ملفات مستقلة خاصة بالتطبـيق. ويعني هذا أنَّ التطبيقات كافةً تتشارك فيما بينها بالبيانات نفسها المخزَّنة في قاعدة البـيانات. وعلى الرغم من مشاركة التطبيقات، والمستفيدين، لنفس قاعدة البيانات؛ فإن نظام إدارة قاعدة البيانات يُمكِّن من إعطاء المستفيدين صلاحياتٍ مُحدَّدة للتعامل مع البيانات المخزَّنة كلٌّ حسب ما يحتاج إليه من بيانات جزئية تُسمَّى «منظور المستفيد» (s View’User).

- تحسين إنتاجية نظم التطبيقات (InCreased ProduCtivity of AppliCations):

يُعَدُّ تحسين الإنتاجية من نظم التطبيقات إحدى المَيْزات الرئيسية لطريقة تطوير التطبيقات باستخدام نظم قواعد البيانات؛ وذلك لسببين رئيسين:

1- بافتراض أنه قد تمَّ تطويرُ التطبيقات التي تقوم بجمع البيانات وحِفظها في قاعدة البيانات؛ فإن المبرمج لأيِّ تطبيق جديد سيقوم بالتركيز على برمجة الوظائف التي سيقوم بها التطبيق قيد التطوير عوضاً عن التركيز على طريقة جمع البيانات وتخزينها في الملفات وطريقة الوصول إليها.

2- توفر غالبية نظم إدارة قواعد البيانات مجموعةً من الأدوات التي تساعد على الإنتاجية، مثل: أدوات «تصميم النماذج» (Form Design) وأدوات «توليد التقارير» (Report Generators)؛ إضافةً إلى لغات عالية المستوى، مثل: «لغة الاستفسار البنائية» ((StruCtured Query Language (SQL) التي تُمكِّن من التعامل مع قاعدة البيانات والوصول إلى البيانات المطلوبة بشكلٍ فعال.

- تقليص تكلفة صيانة البرامج (ReduCed Program MaintenanCe):

إن وَصْف البيانات والمنطق المُستَخدَم للوصول إليها مبنيٌّ داخل التطبيقات نفسها عند استخدام الملفات التقليدية في تطوير النظم؛ ولذلك فإن أيَّ تعديل في «هيئة البيانات» (Data Format) أو طريقة الوصول إليها يتطلَّب تعديل جميع التطبيقات التي تتعامل معها. غير أنَّ نظم قواعد البيانات توفر بعض الاستقلالية بين نظم التطبيقات والبيانات التي تستخدمها بمعنى أن تغيير هيئة البيانات أو طريقة الوصول إليها ليس من الضروري أن ينعكس على جميع التطبيقات التي تتعامل مع هذه البيانات. كذلك هو الحالُ بالنسبة للتعديل على برامج التطبيقات؛ لأنه ليس من الضروري أن ينعكس على البيانات المخزَّنة في قاعدة البيانات. ولذلك؛ فإن هذا يقلص من تكلفة صيانة التطبيقات مقارنةً باستخدام الملفات التقليدية.

1–2 نظم قواعد البيانات:

قاعدة البيانات هي مجموعةٌ من البيانات التي نُظِّمت بشكلٍ متكامل؛ بهدف تلبية احتياجات عددٍ من المستفيدين في المنشأة؛ للقيام بمهام أعمالهم. وتُعَدُّ قاعدة البيانات أساس أيِّ نظام

معلوماتي سواءٌ أكان هذا النظام يدوياً أم آلياً. لذا؛ فإننا نجد أن «نظم قواعد البيانات» (Database Systems) هي المرتكزُ الرئيس الذي تُبنَى عليه نظم المعلومات الآلية. وتتميز نظم قواعد البيانات عن «نظم الملفات» (File Systems) التقليدية التي كانت تُستخدَم لبرمجة التطبيقات مع بداية ظهور الحاسبات الآلية بالمميزات التالية (Efraim et al, 2002; Hoffer et al, 2018; Elmasri and Navathe 2015):

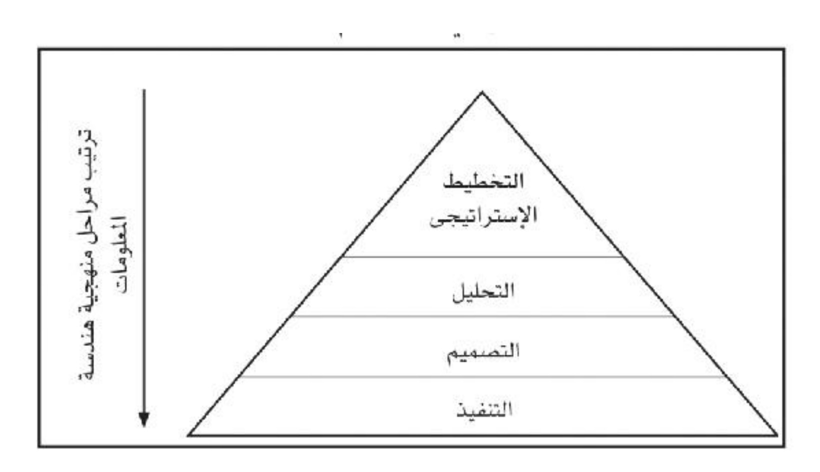
- تكرار محدود للبيانات.
- عدم تعارض البيانات.
- تكامل البيانات.
- مشاركة البيانات.
- سهولة فرض «المقاييس» (Standards).
- زيادة إنتاجية المبرمجين.
- سهولة فرض ضوابط أمن وسلامة البيانات.
- الاستجابة للاحتياجات المتغيرة للبيانات.
- استقلالية البيانات.
- انخفاض صيانة برامج التطبيقات.

وفيما يلي شرحٌ مقتضب لعملية تطوير نظم المعلومات باستخدام نظم قواعد البيانات.

1-2-1 تطوير نظم المعلومات:

تُعَدُّ منهجية «هندسة المعلومات» من أنجح المنهجيات في تطوير نظم المعلومات؛ وذلك لكونها تعطي نظرةً شاملةً للمنشأة وتمَكِّن من تطويع تقنية المعلومات لخدمة أهدافها وسياساتها بكفاءة عالية. ويعني هذا تمكين المنشأة من التعرف على «عوامل نجاحها الحرجة» (CritiCal SuCCess FaCtors)، والعمل على مكننتها بهدف إعطائها الميزة التنافسية وتحسين أدائها.

وترتكز منهجية هندسة المعلومات على أربع مراحل تأخذ شكلاً هرمياً (Martin, 1989)، كما هو مبين في الشكل رقم (1-3).



شكل رقم (1–3): هرمية تطوير نظم المعلومات في المنظمة وفق منهجية هندسة المعلومات.

تأتي في أعلى الهرم، المُمثَّل في الشكل السابق، مرحلة التخطيط الإستراتيجي للمعلومات التي تهتم بـالمعلومات الإستراتيجية للمنشأة، مثل: أهداف المنشأة، وعوامل نجاحها الحرجة. وتتمثل مرحلة التخطيط الإستراتيجي في وَضْع خطة تكفل تكامل نظم المعلومات وجداولها الزمنية التي تحقق أهداف المنشأة. ويعني هذا أن التخطيط الإستراتيجي لنظم المعلومات لا بد أن يرتبط بأهداف المنشأة، وأن يكون بـعيد المدى لكي يتحقق نجاحه. كما يتم تحديد دور الإدارة العليا ومدى التزامها نحو تطوير نظم المعلومات؛ بحيث يجب على الإدارة العليا أن تتبنَّى نظم المعلومات عن قناعة وأن تلتزم بالتعامل معها باعتبـارها أحد الموارد الأساسية للمنشأة. وبناءً على ذلك؛ يتم تحديد السياسات والخطط والبـرامج التي تضمن تنفيذ الخطة الإستراتيجية للمعلومات. وينتج عن هذه المرحلة تحديدٌ للتطبيقات التي تحتاج إليها المنشأة وتوزيعها على «مناطق عمل» (Business Areas)؛ حتى يسهل التعامل معها في أثناء إجراء عملية التحليل المفصَّل والدقيق للمهامِّ والبـيانات. ويعني هذا أن مرحلة التخطيط الإستراتيجي تُمكِّن من إعطاء نظرة عامة عن المنشأة من حيث مهامها، وبـياناتها، واحتياجاتها المعلوماتية، وهيكلها التنظيمي.

تلي مرحلة التخطيط الإستراتيجي للمعلومات مرحلةُ التحليل لأعمال القطاعات المختلفة؛ إذ يتمُّ في هذه المرحلة النظر في مهام كل قطاع والبيانات الخاصة به. كما يتم في هذه المرحلة تحديدُ الاحتياجات من التطبيقات وقواعد البـيانات.

المرحلة الثالثة من منهجية هندسة المعلومات هي مرحلة التصميم؛ حيث يتم في هذه المرحلة تصميمُ النظم التطبيقية وقواعد البيانات للقطاعات المختلفة.

أما المرحلة الرابعة، والتي تمثل قاعدة الهرم في منهجية هندسة المعلومات، فهي مرحلة التنفيذ (أو الإنجاز). في هذه المرحلة يتمُّ تحديد أشكال البيانات، وكتابة برامج التطبيقات.

تجدر الإشارة هنا إلى أن المرحلتين الأولى والثانية تُعَدَّان مستقلتين عن طبيعة الأجهزة والنظم، في حين أن المرحلتين الثالثة والرابعة تعتمدان على طبيعة الأجهزة والنظم المستخدمة. ونظراً لأهمية مرحلة التخطيط الإستراتيجي لنظم المعلومات؛ فإننا نستعرض هذه المرحلة، بشكلٍ مقتضب، فيما يلي.

1-2-1-1 التخطيط الإستراتيجي لنظم المعلومات:

تهدف عملية التخطيط الإستراتيجي لنظم المعلومات إلى المواءمة بين التقنيات المعلوماتية المتوفرة وإستراتيجيات العمل المتبعة في المنظمة. وتُعَدُّ هذه المواءمة مهمةً؛ حتى تتمَّ الاستفادة العظمى من الاستثمارات في النظم المعلوماتية والتقنيات. وتمثل عملية التخطيط أساس عملية الانتقال من العمل اليدوي المعمول به في المنظمة إلى العمل المميكن (أو المحوسب). وتتكون عملية (أو مرحلة) التخطيط في منهجية هندسة المعلومات من ثلاث خطوات (Martin, 1989; Hoffer et al, 2018)، وهي كما يلي:

1- تحديد عوامل التخطيط الإستراتيجي: تتمثل عوامل التخطيط الإستراتيجي في أهداف المنظمة، وعوامل نجاحها الحرجة، ومكامن (أو مصادر) المشكلات. وتهدف عملية تحديد هذه العوامل إلى تطوير إطار لعملية التخطيط يتمُّ فيه الربط بين خطط النظم المعلوماتية وخطط العمل الإستراتيجية للمنظمة. ويحتوي الجدول رقم (1-1) على أمثلة لعوامل التخطيط الإستراتيجي الممكنة في إحدى الجامعات الأهلية.

جدول رقم (1-1): أمثلة لعوامل التخطيط الإستراتيجي.

عوامل التخطيط الإستراتيجي	أمثلة

الأهداف	زيادة عدد الخريجين بمعدل 5%.
	استقطاب كفاءات متخصصة ضِمن أعضاء هيئة التدريس بالجامعة.
عوامل النجاح الحرجة	ارتفاع المستوى الأكاديمي لخريجي الجامعة.
	ارتفاع مستوى الإنتاج العلمي لأعضاء هيئة التدريس.
	زيادة إنتاجية أعضاء هيئة التدريس.
مكامن (أو مصادر) المشكلات	التوقعات غير الصحيحة لأعداد الطلبة المقبولين في الجامعة.
	زيادة مستوى المنافسة مع الجامعات الأهلية الأخرى.

وتساعد العوامل السابقة إداريي نظم المعلومات في وَضْع الأولويات التي تلبي متطلبات المستفيدين من النظم المعلوماتية الجديدة، ومن ثمَّ تطوير قواعد البيانات في المنظمة. فعلى سبيل المثال: مصدر مشكلة التوقعات غير الصحيحة لأعداد الطلبة المقبولين في الجامعة قد يدفع إداريي النظم المعلوماتية إلى توفير المزيد من البيانات التاريخية عن الطلبة المقبولين في الجامعة، ووَضْع المزيد من البيانات التي تنتج من دراسات أعداد خريجي الثانوية العامة، وكذلك البيانات التي تنتج عن احتياجات سوق العمل من خريجي الجامعات.

2- التعرُّف على مكونات (أو وحدات) التخطيط: التعرف على مكونات (أو وحدات) التخطيط في مجال عمل المنظمة، والذي يقوم بتقييد عملية تحليل النظم وتحديد المواقع التي تحدث فيها التغييرات. ويوجد خمسة مكونات (أو وحدات) للتخطيط، وهي كما يلي (Hoffer et al, 2018):

- الوحدات التنظيمية (Organizational Units) التي تتمثل في الإدارات والأقسام المختلفة للمنظمة.

- المواقع التنظيمية (Organizational LoCations) التي تتمثل في المواقع المختلفة التي تدور فيها الفعاليات المختلفة لعمل المنظمة.

- وظائف المنظمة (Business FunCtions) التي تتمثل في مجموعات من العمليات المرتبطة مع بعضها لمساندة مهامِّ المنظمة. وتجدر الملاحظة إلى أن وظائف المنظمة تختلف عن الوحدات التنظيمية؛ إذ إن وظيفة ما قد يتم القيام بها من قِبل أكثر من وحدة تنظيمية واحدة. فعلى سبيل المثال: عملية تسجيل نتائج الدارسين في إحدى الجامعات؛ قد يتم القيامُ بها من خلال القسم الذي يتبعه كل دارس بالإضافة لإدارة (أو قسم) التسجيل في إدارة القبول والتسجيل في الجامعة.

- أنواع الكينونات التي تتمثل في تصنيفٍ للبيانات وفق الأشخاص، والأماكن، والأشياء التي تتعامل معها المنظمة أو تُدار من قِبلها.

- النظم المعلوماتية وتتمثل في التطبيقات البرمجية والإجراءات التي تتعامل مع مجموعات من البيانات.

3- تطوير نموذج المنظمة (Developing an Enterprise Model): يتكون النموذج المفصَّل للمنظمة من تفكيكٍ وظيفيٍّ لكلِّ وظيفة رئيسية تقوم بها المنظمة، ونموذج لبيانات المنظمة، ومصفوفات تخطيط مختلفة.

والتفكيك الوظيفي (FunCtional DeComposition): هو عملية تجزئة لوظائف المنظمة لمستويات أكثر تفصيلاً. وتُعَدُّ عملية التفكيك الوظيفي من العمليات التقليدية التي تُستخدَم في تحليل النظم؛ حتى يتمَّ تبسيط المشكلة، ويتم تركيز الانتباه على حلِّها، وللتعرُّف على مكوناتها.

أمَّا نمذجة بيانات المنظمة؛ فيتم توصيفها باستخدام ترميز معين، مثل: نموذج «كينونة-علاقة»، الذي يمثل محور الفصلين الثاني والثالث من الكتاب. وإضافةً للنموذج ذي الطابع الرسومي؛ يحتوي النموذج المفصَّل لبيانات المنظمة على توصيفٍ لكلِّ كينونة تمثل جزءاً من

بيانات المنظمة، وعلى قواعد العمل المتبعة في المنظمة التي تحكم صحة البيانات. وقواعد العمل هي عبارات لغوية تصف الأحكام والنظم واللوائح، وأيَّة اعتبارات مرعية أخرى تُسيِّر العمل في المنظمة. كما يُبيِّن نموذج بيانات المنظمة أيضاً العلاقات التي تربط بين الكينونات المختلفة في مخطط نموذج بيانات المنظمة.

أمَّا العلاقات الأخرى بين وحدات التخطيط؛ فيتمُّ أخذها بعين الاعتبار أيضاً في أثناء نمذجة المنظمة. ومن الأنماط المتعارف عليها في تمثيل مثل هذه العلاقات، مصفوفات وحدات التخطيط. وتقوم المصفوفات بتوفير وظيفة مهمة، هذه الوظيفة هي إظهار المتطلبات التي تحتاج إليها المنظمة من النظم المعلوماتية دون الحاجة إلى نمذجة قاعدة بيانات المنظمة التي يجب استخدامها مع هذه النظم. كما تساعد المصفوفات في كثيرٍ من الأحيان في وضع أولويات تطوير النظم المعلوماتية، وترتيب عمليات التطوير، وجدولة عمليات التطوير؛ من خلال نظرة شاملة لاحتياجات المنظمة من النظم المعلوماتية. ويُمْكِن استخدام عددٍ من مصفوفات التخطيط التي من بينها، ما يلي:

- **الموقع - الوظيفة:** يحدِّد هذا النوع من المصفوفات كلَّ موقعٍ يقومُ بـممارسة مهمة (أو وظيفة) ما.

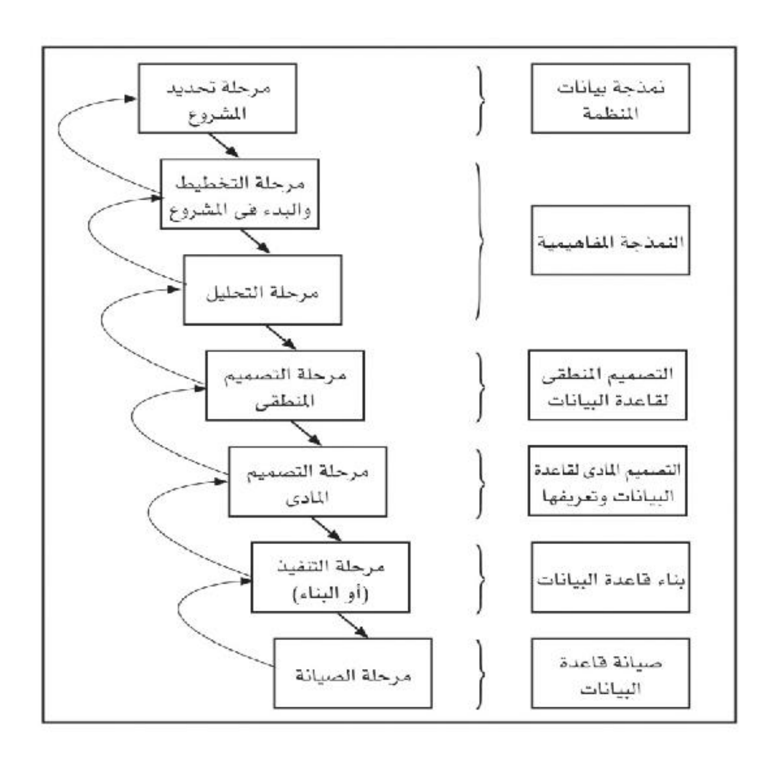
- **الوحدة - الوظيفة:** يحدِّد هذا النوع من المصفوفات كلَّ وحدةٍ إدارية مسئولة عن (أو تقوم بـممارسة) مهمة (أو وظيفة) ما.

- **النظام المعلوماتي - الكينونة:** يوضِّح هذا النوع من المصفوفات تفاعل كلِّ نظام معلوماتي مع الكينونات التي تمثل بيانات المنظمة؛ إذ توضِّح النظم المعلوماتية المسئولة عن إنشاء، أو استرجاع، أو تحديث، أو حذف كلِّ نوعٍ من أنواع البيانات المتوفرة في المنظمة.

وبعد تحديد مناطق العمل والنظم المعلوماتية التي تحتاجها كلُّ واحدة من مناطق العمل؛ يتم تطوير هذه النظم وفق المنهجية الأكثر انتشاراً، وهي «دورة حياة تطوير النظم المعلوماتية»، التي نوجزها كما يلي.

2-1-2-1 دورة حياة تطوير النظم المعلوماتية (System Development Life CyCle (SDLC)):

تُعَدُّ منهجية تطوير النظم المعلوماتية وفق ما يُعرَف بـ «دورة حياة تطوير النظم المعلوماتية» الطريقة التقليدية لتطوير النظم المعلوماتية. وتتكون هذه المنهجية من خطوات متكاملة يتم اتباعها لتوصيف، وتطوير، وصيانة، واستبدال النظم المعلوماتية. ويمكن تصوُّر هذه المنهجية على أنها مجموعةٌ من الخطوات التي تصب من واحدةٍ إلى أخرى، كما هو مُبيَّن في الجزء الأيسر من الشكل رقم (1-4).



شكل رقم (1–4): خطوات تطوير التطبيقات وفق منهجية «دورة حياة تطوير النظم المعلوماتية».

ويُبيِّن الجدول رقم (1-2) الغرض (أو الهدف) من كلِّ مرحلة ومخرجات المرحلة.

جدول رقم (1-2): غرض كلِّ مرحلة من مراحل «دورة حياة تطوير النظم المعلوماتية» ومخرجاتها.

المرحلة	الغرض منها	مخرجاتها
تحديد المشروع	التعرف على دواعي تطوير نظام معلوماتي جديد أو تحسين نظام قائم.	اعتماد تصميم وتطوير نظام معلوماتي لتحسين أداء المنظمة في جانب من الجوانب التي تقوم بها ضمن مهامها.

التخطيط والبدء في المشروع	التعرف على الكيفية التي سيسهم فيها النظام المعلوماتي في تحسين الوضع القائم.	اعتماد دراسة التغييرات الواجب إجراؤها على نظام قائم أو تطوير نظام معلوماتي جديد.
التحليل	التحليل التفصيلي للنظام؛ بهدف تحديد المتطلبات وهيكلتها واختيار البدائل المناسبة لخصائص النظام.	مواصفات النظام الوظيفية التي تلبي احتياجات المستفيدين، ويمكن تطويرها وبناؤها.
التصميم المنطقي	هيكلة كافة متطلبات النظام.	مواصفات وظيفية تفصيلية للبيانات والنماذج، والتقارير، والشاشات، وقواعد معالجة البيانات.
التصميم المادي	تطوير المواصفات التنظيمية والتقنية في المنظمة.	شراء التقنيات التي يتطلبها النظام، وتصميم هياكل البيانات وبرامج النظام.
التنفيذ (أو البناء)	كتابة برامج النظام، وبناء الملفات التي ستحتوي على بيانات النظام، وتركيب واختبار النظام، وتدريب المستفيدين، وتوثيق النظام.	برامج تعمل بشكلٍ سليمٍ وفق مواصفات النظام، ووثائق النظام، وأية مواد مستخدمة لتدريب المستفيدين.
الصيانة	مراقبة عمل وأداء النظام وفوائده، والعمل على إصلاح الأخطاء فيه، وتحسين أدائه.	المراقبة الدورية للنظام؛ للتأكد من عمله بشكلٍ سليم وتلبيته لاحتياجات المنظمة.

ويتشابه الشكل رقم (1-4)، الذي يمثل دورة حياة تطوير النظم المعلوماتية، مع «الشلال المائي» (Waterfall)؛ إذ إن كلَّ خطوة (أو مرحلة) تصب في الخطوة (أو المرحلة) التالية. غير أن هذه الخطوات قد لا تكون منفصلةً تماماً على أرض الواقع؛ فبعض هذه الخطوات قد يتطابق في بعض أجزائها من حيث وقت التنفيذ؛ وذلك في حال إمكانية تنفيذها بشكل متزامنٍ. كما أنه بالإمكان الرجوع إلى خطواتٍ سابقة في الشلال، فيما يشبه التغذية المرتدة؛ وذلك عند ضرورة الرجوع إلى خطوات سابقة لإعادة النظر في القرارات والاعتبارات التي تمَّ اتخاذها في تلك الخطوات.

ونظراً لأن تطوير قواعد البيانات يتم في مراحل موازية لعملية تطوير النظم المعلوماتية؛ فإننا نلخص خطوات تطوير قواعد البيانات المقابلة لخطوات تطوير النظم المعلوماتية فيما يلي.

1-2-1-2-1 عملية تطوير قاعدة البيانات:

يوضِّح الشكل السابق (شكل رقم (1-4))، الذي يمثل دورة حياة النظم المعلوماتية، في جانبه الأيمن، الخطوات المتعلقة بتطوير قاعدة البيانات في كلِّ مرحلة من مراحل تطوير النظام المعلوماتي. وفيما يلي شرحٌ لهذه الخطوات:

- **نمذجة المنظمة (Enterprise Modeling):** تبدأ مرحلة تطوير قاعدة البيانات في أثناء مرحلة نمذجة المنظمة، التي تُعَدُّ جزءاً من مرحلة اختيار وتحديد المشروع. أمَّا مرحلة نمذجة المنظمة، كما أسلفنا أعلاه؛ فتبدأ في مرحلة التخطيط الإستراتيجي للمنظمة. وفي مرحلة نمذجة المنظمة، تتم مراجعة قواعد البيانات المتوافرة في المنظمة، وتحليل طبيعة منطقة العمل التي يتبعها مشروع النظام المعلوماتي قيد التطوير، ويتمُّ توصيف البيانات التي يحتاج إليها النظام بشكلٍ عام دون الخوض في تفاصيل البيانات.

- **النمذجة المفاهيمية (ConCeptual Data Modeling):** يتمُّ في هذه المرحلة تحليلُ المتطلبات العامة من البيانات التي يحتاج إليها النظام المعلوماتي. وتتم هذه العملية ضمن خطوتين: تتمُّ الخطوة الأولى بالتزامن مع مرحلة التخطيط والبدء في مشروع النظام المعلوماتي؛ حيث يتمُّ وضع تصور للبيانات التي يحتاج إليها النظام باستخدام نموذج «كينونة - علاقة» مبسط، وتحديد قواعد البيانات الموجودة فعلياً وتحتوي على بيانات

قد يتعامل معها النظام الجديد. ويتم توصيف البيانات في هذه المرحلة والعلاقات فيما بينها بشكلٍ مُقتضب عالي المستوى دون الدخول في التفاصيل الدقيقة لها. أما الخطوة الثانية فتتمُّ بالتزامن مع مرحلة تحليل المشروع، وتهدف إلى إنتاج نموذج تفصيلي للبيانات يُحدِّد كلَّ البيانات التي يتعامل معها النظام. ويحتوي النموذج التفصيلي على جميع أصناف (أو فئات) البيانات، وجميع حقول أصناف البيانات، وتمثيل لكل العلاقات التي تربط البيانات بعضها ببعض، وتحديد كل قواعد العمل التي تُعنَى بتكامل (أو صحة) البيانات. ويحتوي الفصل الثاني من هذا الكتاب على شرحٍ مُفصَّلٍ

للمكونات الأساسية لنموذج البيانات «كينونة - علاقة» (Entity-Relationship Model) الذي يُعَدُّ أكثر نماذج البيانات المفاهيمية شيوعاً. أما الفصل الثالث؛ فيحتوي على شرحٍ لأهمِّ مكونات نموذج «كينونة - علاقة» المطوَّر الذي يساعد على النمذجة المفاهيمية للبيانات عندما تكون أكثر تعقيداً في العلاقات فيما بينها.

- **التصميم المنطقي لقاعدة البيانات (LogiCal Database Design):** يتم في هذه المرحلة تحويل النموذج المفاهيمي إلى النموذج العلاقي بناءً على نظرية قواعد البيانات العلاقية، وباستخدام شكلٍ قياسي يُسمَّى «العلاقات». وكلما تمَّ تصميمُ أحد برامج النظام المعلوماتي؛ تتمُّ مراجعة تفصيلية للعمليات التي تتفاعل مع قاعدة البيانات، والتقارير، والشاشات التي يحتويها البرنامج؛ بهدف التعرف على كل البيانات التي يتفاعل معها النظام المعلوماتي وطبيعة هذه البيانات. وتوفر مثل عملية المراجعة هذه لبرامج النظام نظرةً شموليةً لقاعدة البيانات من الممكن أن تؤدي إلى إعادة النظر في بعض جوانب النموذج المفاهيمي الذي تمَّ تصميمه في المرحلة السابقة، وتعديلها حسب المعطيات الجديدة. وبعد ذلك تتمُّ عملية تحويل مواصفات البيانات التي يتطلبها النظام إلى علاقات جيدة البناء تحتوي على أقل قدرٍ من تكرارية البيانات التي تؤدي إلى أخطاء التعديل على قاعدة البيانات. وتُستَخدَم في هذه العملية قواعدُ مُستمدَّة من نظرية قواعد البيانات العلاقية تُسمَّى في مجملها عملية «التطبيع» (Normalization). ويشرح الفصل الخامس من الكتاب التصميمَ المنطقيَّ لقواعد البيانات العلاقية، في حين يشرح الجزء الأول من الفصل السادس مفهوم تطبيع العلاقات ومستويات تطبيعها.

- **التصميم المادي لقاعدة البيانات وتعريفها (PhysiCal Database Design):** في هذه المرحلة يتم تحديد طريقة تنظيم قاعدة البيانات على وحدات تخزين الحاسب الآلي، وهي الأقراص الصلبة عادةً، كما يتم تعريف الهياكل المادية لإدارة قاعدة البيانات. وتهدف هذه المرحلة، بشكلٍ عام، إلى التصميم الجيد لقاعدة البيانات على وحدات التخزين وتصميم الهياكل المناسبة لها؛ بحيث يمكن للمستفيدين والنظم المعلوماتية أن تتعامل مع قاعدة البيانات بشكلٍ فعَّال؛ من حيث الأداء بالإضافة إلى تأمين السرية في التعامل معها. ويعني هذا أن التصميم المادي لقاعدة البيانات يجب أن يتمَّ بتناسقٍ كاملٍ مع مراحل التصميم المادية الأخرى للنظام المعلوماتي، مثل برامج النظام، وبشكلٍ يتوافق مع مكونات النظام الحاسوبي المُستخدَم، مثل نظام التشغيل،

وشبكة الاتصالات المستخدمة. ويشرح الجزء الثاني من الفصل السادس بعض مفاهيم التصميم المادي والهياكل المُستخدَمة في عملية التصميم المادي لقواعد البيانات.

- بناء قاعدة البيانات (Database Implementation): في مرحلة بناء قاعدة البيانات تتمُّ كتابة، واختبار، وتركيب البرامج التي ستتعامل مع قاعدة البيانات. وقد تتم كتابة هذه البرامج باستخدام لغات البرمجة العامة (مثل: كوبول، وسى، وجافا)، أو لغات معالجة خاصة بقواعد البيانات، مثل: «لغة الاستفسار البنائية» (SQL)، أو أية لغة خاصة لكتابة التقارير وإظهار الشاشات والتي قد تحتوي على بعض الرسومات المعبرة. كما يتمُّ في هذه المرحلة إنهاءُ عملية توثيق تصاميم قاعدة البيانات، وتدريب المستفيدين منها. أمَّا الخطوة الثانية من هذه المرحلة؛ فهي عملية تعبئة البيانات في قاعدة البيانات، وتتم هذه الخطوة من خلال تحويل البيانات المتوفرة في ملفات النظم قيد الاستخدام إلى صيغة قياسية (مثل: «النظام الثنائي» (Binary System) أو ASCII أو Unified Code)، ثم تعبئتها في قاعدة البيانات حسب صيغة البيانات المُستَخدَمة في قاعدة البيانات. أمَّا في حالة عدم توفر أية بيانات للنظام الجديد من نظم سابقة قيد الاستخدام؛ فتتم عملية تعبئة البيانات حسب توفرها للنظام وإدخالها أولاً بأول.

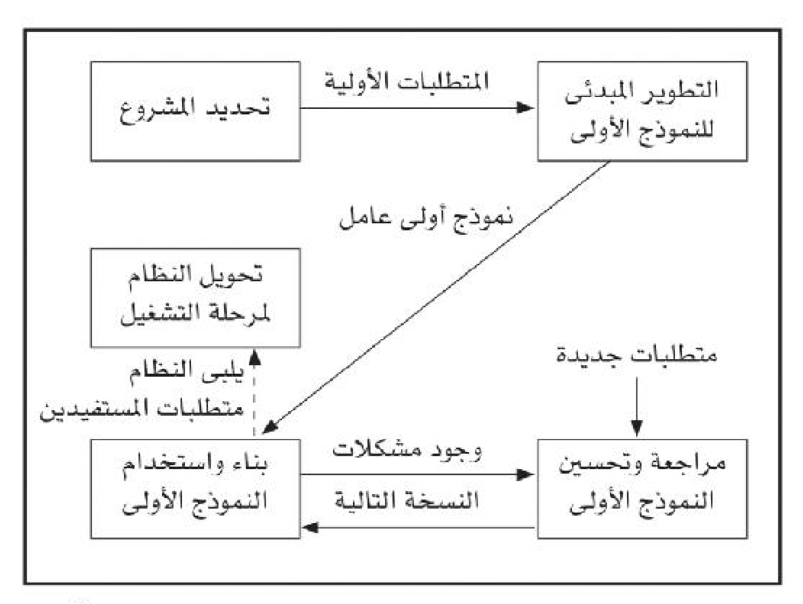
- صيانة قاعدة البيانات (Database MaintenanCe): في مرحلة الصيانة تتم مراقبة قاعدة البيانات؛ وذلك لكونها تتصف بارتقائها وتطوُّرها المستمر نتيجة لعمليات الحذف، والتعديل، والإضافة لهياكل بياناتها؛ حتى تتناسب مع التغيرات المستمرة في بيئة العمل؛ أو لتصحيح الأخطاء في تصميم قاعدة البيانات، أو لتحسين أداء معالجة البيانات المخزنة في قاعدة البيانات. ويمكن النظرُ لهذه المرحلة على أنها مرحلة تطوير مقتضبة لقاعدة البيانات تحتوي على مراحل النمذجة المفاهيمية، والتصميم المنطقي، والتصميم المادي، والبناء؛ حتى يمكن التعامل مع التغيرات المستمرة لقاعدة البيانات. ويمكن النظر لمرحلة الصيانة على أنها عملية تطوير لقاعدة البيانات ضمن فترات مراجعة يتمُّ فيها النظرُ في التغيرات الحادثة في بيئة العمل وما تتطلبه هذه التغيرات من تطوير لقاعدة البيانات.

1-2-1-3 طرق التطوير البديلة لنظم المعلومات:

على الرغم من أن تطوير النظم المعلوماتية وفق منهجية «دورة حياة تطوير النظم المعلوماتية» التي تحتوي على عددٍ من الخطوات و«نقاط التحقق» (CheCkpoints)؛ لضمان تطوير النظم بدرجة عالية من الدقة والصحة في نتائجها وتلبيتها لمتطلبات المستفيدين منها، غير أن هذه المنهجية يتم انتقادها عادةً؛ بسبب طول الفترات الزمنية التي تحتاج إليها لإنتاج النظم المعلوماتية، خاصةً أن النظم التي يتم تطويرها وفق هذه المنهجية لا تكون جاهزةً للعمل إلا في المرحلة الأخيرة من هذه المنهجية. ولهذا السبب؛ فإن العديد من المنظمات تستخدم طرق تطوير بديلة لنظم المعلومات تتسم بالتطوير السريع لنظم التطبيقات. وتسمح هذه الطرق بالتطوير السريع للنظم من خلال عملية متكررة لمراحل التحليل، والتصميم، والبناء؛ حتى يصبح النظام مُلبياً لاحتياجات متطلبات المستفيدين. وتعمل طرق التطوير السريعة هذه أفضل ما يكون عندما تكون قواعد البيانات التي يتطلبها النظام قيد التطوير موجودةً فعلياً على أرض الواقع. ومن ثم؛ فإن النظام الجديد يتميز بكونه نظام استرجاع للبيانات عوضاً عن كونه نظام تحديث أو إدخال للبيانات، ويتطلب مراجعةً لهياكل قواعد البيانات.

1-2-1-3-1 النموذج الأوَّلي (**Prototyping**):

تُعَدُّ طريقة «النموذج الأوَّلي» واحدةً من أكثر طرق التطوير السريع لنظم المعلومات شيوعاً. وطريقة النموذج الأوَّلي؛ هي عملية تطوير للنظم المعلوماتية يتمُّ فيها تحويل متطلبات المستفيدين إلى نظام عامل؛ بحيث تتم مراجعته بشكلٍ متكررٍ بين محللي النظام والمستفيدين منه؛ حتى يتمَّ الوصول إلى نسخة توافق متطلبات المستفيدين. ويوضِّح الشكل رقم (1-5) خطوات تطوير النظم المعلوماتية وفق طريقة النموذج الأوَّلي.



شكل رقم (١-٥): خطوات تطوير النظم المعلوماتية وفق طريقة النموذج الأوَّلي.

وبينما يُوضِّح الشكل رقم (1-5) مراحل تطوير النظم المعلوماتية وفق طريقة النموذج الأوَّلي؛ فإن الجدول رقم (1-3) يوضِّح الفعاليات المصاحبة لهذه المراحل، والتي تتعلق بتطوير قاعدة البيانات بشكل تقريبي.

جدول رقم (1-3): الفعاليات المتعلقة بقواعد البيانات المصاحبة لمراحل تطوير النظم المعلوماتية وفق طريقة النموذج الأوَّلي.

مرحلة تطوير النظام المعلوماتي	الفعاليات المتعلقة بقواعد البيانات
تحديد المشروع	نمذجة مفاهيمية: - تحليل متطلبات النظام. - تطوير نموذج مبدئي للبيانات التي يتطلبها النظام.
التطوير الأوَّلي للنموذج المبدئي	1- التصميم المنطقي لقاعدة البيانات: - تحليل تفصيلي لمتطلبات النظام. - تحويل النموذج المفاهيمي إلى النموذج العلاقي. - تطبيع العلاقات.

	2- التصميم المادي لقاعدة البيانات: - تعريف محتويات قاعدة البـيانات. - تنظيم محتويات قاعدة البـيانات مادياً. - تصميم البرامج التي تتعامل مع قاعدة البـيانات.
بناء واستخدام النموذج الأوَّلي	بناء قاعدة البيانات: - كتابة التعليمات التي تتعامل مع قاعدة البيانات ضِمن بـرامج النظام. - تعبئة البيانات في قاعدة البيانات من مصادر البيانات المتوفرة (في النظم القائمة).
مراجعة وتحسين النموذج الأوَّلي	صيانة قاعدة البيانات: - تحليل قاعدة البيانات؛ للتأكد من تلبيتها لمتطلبات النظام المعلوماتي. - تصحيح الأخطاء في قاعدة البيانات.
تحويل النظام لمرحلة التشغيل	صيانة قاعدة البيانات: - تحسين أداء قاعدة البـيانات. - تصحيح الأخطاء في قاعدة البـيانات.

تتمُّ عملية النمذجة المفاهيمية لقاعدة البيانات، وبشكلٍ مقتضب، عندما يتمُّ تحديد النظام المعلوماتي الذي يجب تطويره. وفي مرحلة التطوير المبدئي للنموذج الأوَّلي يتم تصميم الشاشات والتقارير التي تلبي احتياجات المستفيدين، وفي الوقت نفسه، التعرف على أية متطلبات إضافية يحتاج إليها المستفيدون من قاعدة البيانات؛ إضافةً إلى تعريف قاعدة البـيانات. وعادةً ما تكون قاعدة الـبيانات الخاصة بالنظام قيد التطوير نسخة تتألف من أجزاء قواعد بيانات متوفرة في المنظمة وقيد العمل بها، مع إمكانية احتوائها على بيانات جديدة.

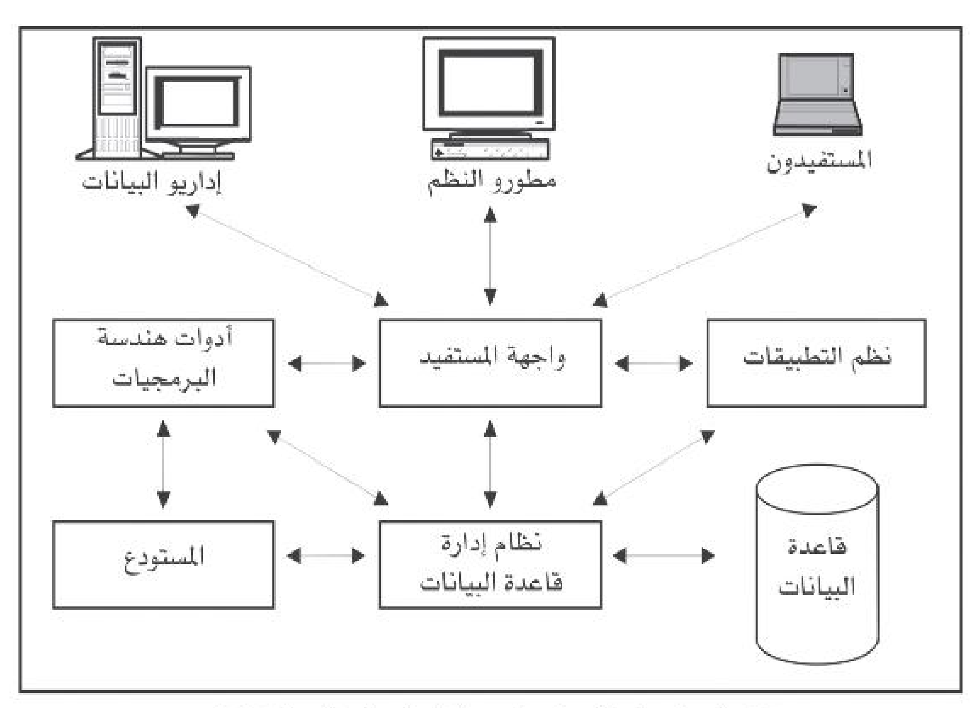
وبعد تطوير النموذج المبدئي والبدء في استخدامه؛ ينتقل النظام إلى حلقة تتكرَّر فيها مرحلة البـناء ومرحلة المراجعة. وعند بداية انتقال النظام لهذه الحلقة، تكون جوانب السِّرية والتكامل في النظام أقل ما يمكن؛ لأن التركيز كان منصباً على الحصول على نموذج أوَّلي عامل. وبعد أن

يصل النظامُ إلى وَضْعٍ مستقر يُلبي غالبية متطلبات المستفيدين، نتيجةً لمروره بعددٍ من مراحل المراجعة والبناء، ويُقرِّر كلٌّ من المستفيدين ومُطوِّري النظام نقله للمرحلة الإنتاجية (أو التطبيقية)، يدخل النظام في مرحلة الصيانة، ويبقى في هذه المرحلة ما بقي النظام قيد الاستخدام.

وكما يُلاحَظ من خلال الشرح السابق؛ فإن طرق التطوير السريعة تمكِّن من الوصول إلى نظام عامل بأقل وقتٍ ممكن؛ ولكنه يُلبي أقلَّ قدرٍ ممكن من احتياجات المستفيدين. أما منهجية دورة حياة النظم المعلوماتية فتُلبي أكبر قدرٍ ممكنٍ من احتياجات المستفيدين؛ ولكنها تتطلب وقتاً كبيراً للوصول إلى نظام عامل. ولذلك؛ فإن الاختيار بين الطريقتين يعتمدُ بحَدٍّ كبيرٍ على طبيعة النظام المعلوماتي قيد التطوير، وخاصةً درجة تلبيته لاحتياجات المستفيدين في صورته الأولى، من جانب، والوقت المتوفر لعملية تطوير النظام، من جانب آخر.

1-2-2 مكونات بيئة نظام قاعدة البيانات:

تُعَدُّ البيئة التشغيلية لنظام قاعدة البيانات بيئةً متكاملةً من الأجهزة والبرامج والأفراد الذين يشرفون على إدارة وتشغيل نظام قواعد البيانات. ويوضح الشكل رقم (1-6) المكونات الرئيسية لبيئة نظام قواعد البيانات وعلاقتها ببعضها.



شكل رقم (١-٦): المكونات الرئيسية لبيئة نظام قاعدة البيانات.

وفيما يلي توضيحٌ للمكونات الرئيسية لبيئة قاعدة البيانات:

1- أدوات هندسة البرمجيات المساعدة : هي أدوات آلية مصاحبة لنظام قاعدة البيانات تُستخدَم عادةً في عملية تصميم قواعد البيانات وبرامج التطبيقات.

2- مستودع (Repository): يُعَدُّ المستودع قاعدة معارف مركزية تحتوي على تعريف لجميع البيانات والشاشات والتقارير. كما يحتوي المستودع على معلومات عن جميع مخططات قاعدة البيانات والقيود عليها، وعلى معلومات أخرى يدخل من ضِمنها القرارات التي تمَّ اتخاذها في عملية تصميم قاعدة البيانات، ومقاييس استخدام قاعدة البيانات، ووَصْف للتطبيقات التي ستتعامل مع قاعدة البيانات، ومعلومات عن المستفيدين من قاعدة البيانات. ويمكن الرجوع إلى هذه المعلومات عند الحاجة، سواء من المستفيدين من قاعدة البيانات أم من إداريي قاعدة البيانات. ويمكن تشبيه مستودع المعلومات بكتالوج نظام إدارة قاعدة البيانات (DBMS Catalog) الذي يحتوي على معلومات تفصيلية عن البيانات المخزَّنة في قاعدة البيانات، وطرق استرجاعها (ACCess Paths)، وأماكن وجودها؛ إضافةً إلى معلومات عن حقوق المستفيدين في التعامل معها (Users’ ACCess Information). غير أنَّ المستودع يحتوي على معلومات أكثر تنوعاً من كتالوج نظام إدارة قاعدة البيانات، كما أنه مسخرٌ للتعامل معه بشكلٍ مباشرٍ من قِبل المستفيدين من نظام قاعدة البيانات، من مبرمجين للتطبيقات وإداريين لقاعدة البيانات، عوضاً عن استخدامه من قِبل برامج نظام إدارة قاعدة البيانات الذي هو حال كتالوج نظام إدارة قاعدة البيانات.

3- نظام إدارة قاعدة البيانات (Database Management System (DBMS)): هو نظام برمجي تجاري يُستخدَم لإدارة قاعدة البيانات من حيث تخزين واسترجاع وتحديث البيانات طبقاً لمتطلبات المستفيدين من قاعدة البيانات، كما أنه مسئول عن سلامة البيانات وتكاملها حتى في ضوء التداول المتزامن لها من قِبل المستفيدين من البيانات، وفي حالة حدوث عطل من الأعطال للنظام الحاسوبي الذي يحتوي على قاعدة البيانات.

4- قاعدة البيانات (Database): هي مجموعةٌ من البيانات المترابطة منطقياً فيما بينها تمَّ تصميمها لتلبية الاحتياجات المعلوماتية لمجموعةٍ من المستفيدين في المنظمة. وتحتوي قاعدة

البيانات على البيانات الفعلية الموجودة في المنظمة عوضاً عن وصفها الذي يكون مخزناً في المستودع.

5- برامج التطبيقات (AppliCation Programs): هي مجموعةٌ من البرمجيات تمَّ تطويرها لتستخدم من قِبل المستفيدين من قاعدة البيانات؛ بحيث تمكِّنهم من إدخال البيانات في قاعدة البيانات والتعديل عليها، بالإضافة لمعالجتها للحصول على معلومات تتناسب مع احتياجاتهم.

6- واجهة المستفيد (User InterfaCe): هي مجموعةٌ من اللغات والقوائم (Menus) والأوامر والنماذج (Forms) التي تمكِّن المستفيدين من التعامل مع بقية مكونات النظام وما فيها من أدوات هندسة البرمجيات، وبرامج التطبيقات، ونظام إدارة قاعدة البيانات، والمستودع.

7- إداريو قواعد البيانات (Database Administrators (DBAs)): هم الأشخاص المسئولون عن تصميم قواعد البيانات، ووَضْع سياسات أمنها وسلامتها، واستعادتها لوضعها التشغيلي الطبيعي بعد حدوث الأعطال. ويستخدم إداريو قواعد البيانات نظام إدارة قواعد البيانات، وأدوات هندسة البرمجيات والمستودع في أداء مهامهم.

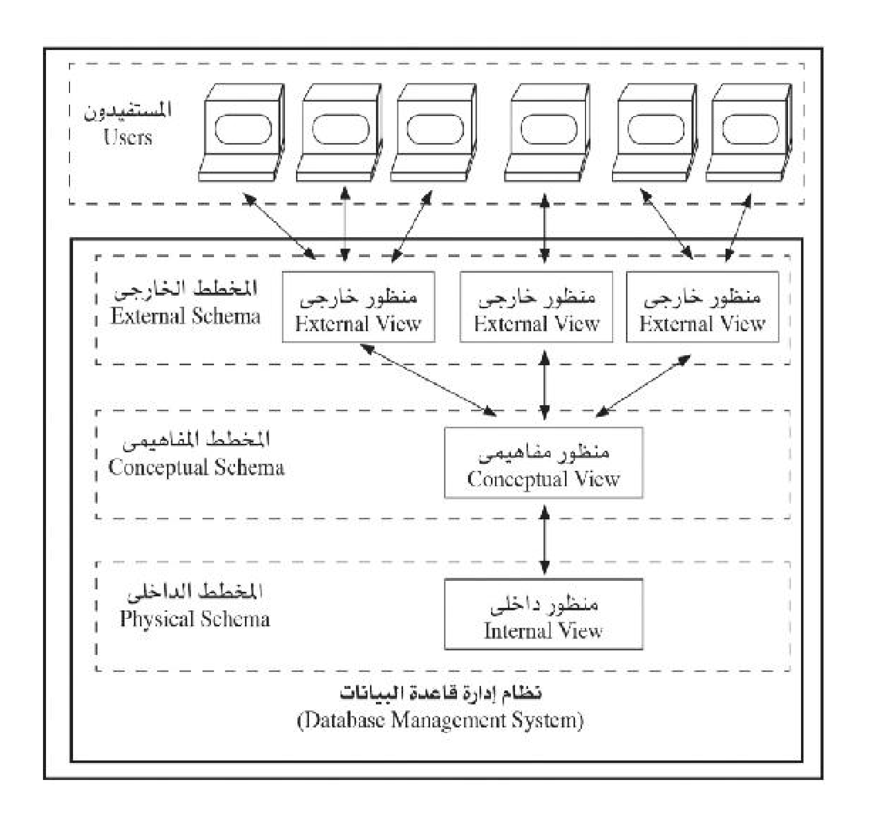
8- مُطوِّرو النظم (Developers ’Systems): هم مُحلِّلو النظم والمبرمجون الذين يقومون بتطوير نظم التطبيقات التي تحتاج إليها المنظمة. وعادةً ما يستخدم مطورو النظم أدوات هندسة البرمجيات في عملية تحليل احتياجات المستفيدين وتصميم النظم.

9- المستفيدون (End Users): هم الأشخاص الذين يتعاملون مع قاعدة البيانات من خلال إدخال البيانات إليها أو إجراء التعديلات عليها، ويطلبون أو يستقبلون معلومات منها.

ويُلاحَظ في الشكل السابق لبيئة نظام قاعدة البيانات أن جميع التعاملات مع قاعدة البيانات المخزَّنة في النظام لا بد أن تمرَّ من خلال نظام إدارة قاعدة البيانات (DBMS)؛ لكونه الجزء المسئول عن سلامة البيانات وتكاملها في جميع الأحوال التشغيلية للنظام.

3-2-1 مستويات التجريد: المنظورات الثلاثة (Levels of AbstraCtion: Three-SChema ArChiteCture):

إن الهدف من وراء بناء قواعد البيانات وفق المنظورات الثلاثة؛ هو الفصل بين برامج التطبيقات التي يستخدمها المستفيدون والبيانات المخزَّنة في قاعدة البيانات. وبهذه الطريقة يتم توصيف البيانات في نظم إدارة قواعد البيانات وفق ثلاثة مستويات من التجريد. ويتكون توصيف البيانات في كل مستوى من هيكل (SChema) للبيانات. ويوضح الشكل رقم (1–7) مستويات التجريد الثلاثة، وعلاقة كلٍّ منها بالأخرى. كما تجدر الإشارة هنا إلى أن المستويات الثلاثة كافة ما هي إلا توصيف لقاعدة البيانات نفسها، ولكنه يتم من خلال منظورات مختلفة.



شكل رقم (1–7): مستويات التجريد في بيئة نظام قاعدة البيانات.

وفيما يلي توضيحٌ لمستويات التجريد الثلاثة الواردة في الشكل (1-7):

1- المنظور الداخلي (Internal View): للمنظور الداخلي هيكل (SChema) يصف التركيبة الداخلية لقاعدة البيانات متضمناً ذلك تفاصيل تخزين البيانات، مثل: مسميات الملفات المخزنة فيها، وأماكن تخزينها على الأقراص الصلبة، وعما إذا كانت البيانات متكررة على مجموعة من الأقراص الصلبة للتمكُّن من استعادتها في حالة الأعطال؛ إضافةً إلى طرق الوصول إليها.

2- المنظور المفاهيمي (ConCeptual View): للمنظور المفاهيمي هيكلٌ (SChema) يصف تركيبة كامل قاعدة البـيانات لمجموعة من المستفيدين. ويخفي المنظور المفاهيمي تفاصيل تخزين الـبيانات في المستوى الداخلي؛ حيث يركِّز على وصف الكينونات، والعلاقات، ونوعية البيانات (Data Types)، والقيود المفروضة على قاعدة البيانات دون الخوض في تفاصيل تخزين البـيانات وطريقة الوصول إليها.

3- المنظور الخارجي (External View): يتكون المنظور الخارجي من مجموعةٍ من الهياكل التي تصف منظورات المستفيدين من قاعدة البـيانات. ولكل مجموعة من المستفيدين منظورها الخاص الذي يهمها من قاعدة الـبيانات. وكل هيكل خارجي يصف جزءاً من قاعدة البيانات يدخل ضمن اهتمام مجموعة محددة من المستفيدين ويخفي ما تبقى من قاعدة البـيانات عن هذه المجموعة.

تجدر الملاحظة إلى أن المنظورات الثلاثة ما هي إلا وَصْفٌ لقاعدة البيانات؛ إذ لا يوجد سوى قاعدة بـيانات واحدة مخزنة في المستوى المادي. ولأن كلَّ مستخدمٍ يتعامل مع منظوره (الخارجي) لقاعدة الـبيانات؛ فإن نظام إدارة قاعدة البيانات يجب أن يترجم أيَّ تعاملٍ من قِبـل المستخدم لما يكافئه من تعامل على المنظور المفاهيمي، ومن ثم ما يكافئه على المنظور الداخلي.

ويُمكِن استخدامُ المنظورات الثلاثة لشرح المقصود باعتمادية البيانات بشكلٍ أوضح، والتي يُقصَد بها في هذه الحالة القدرة على تغيير هيكل البيانات في مستوى ما دون الحاجة إلى تغيير هيكل البـيانات في المستوى الذي يعلوه. ويقودنا هذا إلى تعريف مستويين من عدم اعتمادية الـبيانات، كما يلي:

1- عدم الاعتمادية المنطقية: هي القدرة على تغيير الهيكل المفاهيمي دون الحاجة إلى تغيير المخطط الخارجي أو برامج التطبيقات؛ إذ يمكننا إضافة جدول جديد أو حقل ضمن جدول أو قيد جديد على قاعدة البيانات دون الحاجة إلى تغيير المخططات الخارجية أو التعديل على برامج التطبـيقات. كما أنه يمكننا حذف جدول أو حقل ضمن جدول دون الحاجة إلى تعديل المنظورات كافة أو التطبيقات كافة، وإنما يُكتفَى بتعديل تلك المنظورات والتطبيقات التي ستتأثر بـمثل عملية

الحذف هذه. كما يمكن حذف بعض القيود أو التعديل عليها دون الحاجة إلى تعديل أيٍّ من المنظورات الخارجية، أو أيٍّ من برامج التطبيقات.

2- عدم الاعتمادية المادية: هي القدرة على تعديل الهيكل الداخلي لقاعدة البيانات دون الحاجة إلى تغيير الهيكل المفاهيمي، ومن ثم عدم الحاجة إلى تغيير الهياكل الخارجية كذلك. ومن التغييرات التي قد تطرأ على الهيكل الداخلي إعادة ترتيب بعض ملفات قاعدة البيانات، مثل: إنشاء هياكل جديدة للوصول إلى البيانات (ACCess StruCtures) التي تساعد على الوصول إلى البيانات بفاعلية أكبر.

1-2-4 أنواع قواعد البيانات المتوافرة على المستوى التجاري:

يتوافر حالياً في الأسواق عددٌ كبيرٌ جداً من نظم إدارة قواعد البيانات التي تُراوح مجالات استخداماتها بين «الشخصية» (Personal Databases) القابلة للاستخدام الفردي، مثل تلك التي يمكن تنصيبها على «الحاسبات الشخصية» (Personal Computers)، أو «المساعد الرقمي الشخصي» (Personal Digital AssistanCe)، وصولاً إلى تلك التي يمكن تنصيبها واستخدامها على «الحاسبات المركزية» (Mainframes) التي قد يصل أعداد مستخدميها إلى الآلاف منهم. وعلى الرغم من الاختلافات الجوهرية، في بعض الأحيان، في التقنيات المستخدمة في بناء نظم قواعد البيانات؛ فإنها تجتمعُ فيما تقدِّمه من مَيْزات على نظم الملفات التقليدية بوصفها وسيلةً لتخزين وإدارة البيانات، التي سبق أن تمَّ استعراضها أعلاه. ويُعنَى هذا الكتاب، وبشكلٍ خاص، بنظم قواعد البيانات العلاقية؛ وذلك لكونها الأكثر انتشاراً في وقتنا الراهن على الرغم من وجود نظم إدارة قواعد بيانات مبنية على نماذج أخرى للبيانات، مثل: «الهرمية» (HierarChiCal)، و«الشبكية» (Network)، و«الشيئية» (ObjeCt-Oriented).

3-1 سَرْد تاريخي لتطوُّر نظم قواعد البيانات:

كانت بداية ظهور نظم إدارة قواعد البيانات في الستينيات الميلادية، وهي منذ ذلك الحين تتطور بشكلٍ مستمر. ويستعرض هذا الجزء من الكتاب أهمَّ التطورات في هذا المجال.

- **الستينيات الميلادية:** اعتمدت الغالبية العظمى من التطبيقات في هذه الحِقبة الزمنية على الملفات في بنائها؛ غير أن هذه الحقبة شهدت ظهورَ أولى نظم إدارة قواعد البيانات، وقد تمَّ تصميمها من قِبل شارلز باتشمان (Charles BaChman) الذي كان يعملُ في شركة جنرال إلكتريك (Ramakrishnan and Gehrke, 2003). وقد سُمِّي هذا النظام بمخزن البيانات المتكامل (Integrated Data Store) الذي أصبح فيما بعد أساساً لنموذج البيانات الشبكي (Network Data Model)، كما تمَّ وضع مقاييس لهذا النموذج من خلال مؤتمر عرف بمؤتمر لغات نظم البيانات (ConferenCe on Data Systems Languages (CODASYL)) الذي كان له أثرٌ كبيرٌ في مسار نظم قواعد البيانات عبر الستينيات الميلادية. ولقد حصل باتشمان على أول جائزة تورينغ (ACM Turing Award)، وهي جائزة الحاسب الآلي المكافئة لجائزة نوبل، على أعماله في مجال نظم قواعد البيانات عام 1973م (Ramakrishnan and Gehrke, 2003). وفي أواخر الستينيات الميلادية، طورت شركة أي. بي. إم. (IBM) نظاماً سُمِّي بنظام إدارة المعلومات (Information Management System (IMS)) الذي أصبح فيما بعد أساساً لنموذج البيانات الهرمي (HierarChiCal Data Model) الذي يُعَدُّ نموذجاً بديلاً للنموذج الشبكي.

وكان استخدامُ نظم قواعد البيانات في الستينيات الميلادية محصوراً في التطبيقات التي تغلب عليها ضخامة البيانات، وتعقُّد مهامها، مثل التطبيقات التي أُستخدِمت في مشروع هبوط مركبة الفضاء أبولو (Apollo) على سطح القمر (Hoffer et al, 2018). كما تميَّزت هذه الفترة أيضاً ببداية المجهودات التي تهدف إلى وضع المقاييس المتعلقة بنظم إدارة قواعد البيانات؛ من خلال تشكيل فريق مهمة قواعد البيانات (Data Base Task Group) مع نهاية العقد.

- **السبعينيات الميلادية:** أضحت قواعد البيانات في هذا العِقد واقعاً ملموساً على المستوى التجاري؛ فتمَّ تطويرُ نظم إدارة قواعد البيانات «الهرمية» (HierarChiCal) و«الشبكية» (Network) للاستخدام في تطوير التطبيقات ذات هياكل بيانات معقدة يَصعُب تطويرها باستخدام الملفات التقليدية. وتمثل نظم إدارة قواعد البيانات الهرمية والشبكية التي طُوِّرت في هذه المرحلة الجيلَ الأول لنظم إدارة قواعد البيانات على المستوى التجاري. وقد تمَّ استخدامُ كلا النموذجين بشكلٍ

كبيرٍ في هذه المرحلة. وما زالت هنالك بعض النظم التي تستخدم هذين النموذجين قيد العمل والاستخدام حالياً.

وعلى الرغم من نجاح كلا النموذجين وانتشارهما في هذه المرحلة؛ فإنَّ كليهما كان يعاني بعض المساوئ الجوهرية التي يمكن تلخيصها فيما يلي:

1- صعوبة التنقل بين البيانات: فالبيانات تُخزَّن على هيئة «سجلات» (ReCords) ووسيلة التنقل المُستخدَمة تعتمد، في كلا النموذجين، على الانتقال من سجل للبيانات إلى آخر. ويعني هذا ضرورة كتابة برامج معقدة للإجابة حتى عن أبسط الاستفسارات التي تُجرى على قاعدة البيانات.

2- محدودية الاستقلالية بين البيانات والبرامج التي تتعاملُ معها، ومن ثمَّ؛ فإن البرامج ليست بمعزل عن «هيئة البيانات» (Data Format).

3- لا يُوجَد لأيٍّ من النموذجين أسس نظرية مقبولة بشكلٍ كبيرٍ تُمكِّن من فهم وتحليل أثر التعامل مع البيانات المخزنة في قاعدة البيانات أو نتائج الاستفسارات التي تطبق على قاعدة البيانات.

وقد حَدَت المساوئ التي تشوب كلا النموذجين السابقين بأحد الباحثين في شركة أي. بي. إم. (IBM) ويُدعَى «إدغار كود» (Edgar Codd)، عام 1970م، إلى اقتراح نموذج جديد سُمِّي بالنموذج العلاقي (Codd, 1970). ويمثل النموذج العلاقي الجيل الثاني لنظم إدارة قواعد البيانات.

- الثمانينيات الميلادية: لاقى النموذجُ العلاقي قبولاً كبيراً من المهتمين في نظم إدارة قواعد البيانات، وانتشر استخدامُ هذا النموذج بشكلٍ كبيرٍ على المستوى التجاري. واتسم هذا النموذج ببساطته في تنظيم البيانات، وبزيادة درجة عدم الاعتمادية (أو الارتباط) بين برامج التطبيقات من جانب والبيانات من جانب آخر. وتميَّز بسهولة الاستخدام حتى من قِبل غير المبرمجين؛ إذ إن كافة البيانات والعلاقات فيما بينها تُمثَّل على هيئة جداول بسيطة يمكن فهم محتوياتها والتنقل بينها بسهولةٍ كبيرة. كما صاحب هذا النموذج أسسٌ نظريةٌ تمكِّن من فهم وتحليل الطرق التي يتم التعامل فيها مع البيانات المخزَّنة في قاعدة البيانات.

وأُستخدِم مع هذا النموذج لغات تداول قواعد البيانات العلاقية (مثل SQL) تسمح بمعالجة البيانات في حالة مجموعات أو جداول (عوضاً عن السجلات)، وبحيث لا يحتاج المستفيد أو المبرمج إلى وَصْف مسار البحث عن البيانات في الاستفسارات أو برامج التطبيقات كما هو الحال في نماذج قواعد البيانات الأخرى. وتم تطوير لغة الاستفسار البنائية (SQL) من قِبل شركة أي. بي. إم. لتصبح جزءاً من مشروع نظام إدارة قاعدة البيانات المعروف «بنظام آر» (System R). كما تمَّ وَضْع مقاييس لهذه اللغة بنهاية الثمانينيات الميلادية، وتمَّ تبنِّيها من قِبل معهد المقاييس الوطني الأمريكي (AmeriCan National Standards Institute (ANSI)) والمنظمة الدولية للمقاييس (International Standards Organization (ISO)).

وبناءً على مجهودات «كود» في مجال نظم قواعد البيانات؛ حاز على جائزة تورينغ عام 1981م.

- **التسعينيات الميلادية:** تطوَّرت عمارة الحاسبات الآلية (Computer ArChiteCture) ونظم الاتصالات والشبكات تطوراً كبيراً خلال هذا العِقد. ومع هذا التطوُّر تطورت طبيعة التطبيقات، وظهرت مفاهيم حديثة لم تكن معروفة فيما سبق إلا كضرب من الخيال العلمي المحدود جداً إذا ما قورن بكمية ونوعية التطبيقات والمفاهيم التي ظهرت في هذا العقد. فمع تطوُّر نظم الحاسبات الآلية ونظم الاتصالات والشبكات؛ أصبح بالإمكان معالجة ونقل كميات كبيرة من البيانات، مثل: الرسومات، والصور، ولقطات الفيديو، والصوت. وعليه؛ أصبح بالإمكان تطوير تطبيقات جديدة تلبي احتياجات المفاهيم الحديثة، مثل: التجارة الإلكترونية، والتعليم عن بعد، والحكومة الإلكترونية، والحرب الإلكترونية، على سبيل المثال لا الحصر. ونتيجةً للكمِّ الهائل من البيانات التي تتعامل معها مثل هذه التطبيقات؛ كان لزاماً أن تتطور نظم إدارة قواعد البيانات. وبالفعل تطورت هذه النظم، وأخذت عملية التطوير شكلين رئيسيين: الأول منهما تمثل في نموذج قواعد البيانات الشيئي (ObjeCt-Oriented Data Model) الذي بدأ ظهوره الفعلي في أواخر الثمانينيات الميلادية. وظهر العديدُ من نظم إدارة قواعد البيانات المبنية على هذا النموذج. أما التطور الثاني فتمثل في تحديث بعض منتجي نظم إدارة قواعد البيانات المبنية على النموذج العلاقي لنظمهم؛ بحيث تحتوي على بعض مفاهيم النموذج الشيئي. وأصبحت هذه النظم معروفةً بقواعد البيانات العلاقية الشيئية (ObjeCt-Relational Databases).

- **بداية القرن الحادي والعشرين وما بعد:** دخلت نظم إدارة قواعد البيانات في منظومة شبكة الإنترنت؛ فأصبح الكثيرُ من مواقع الإنترنت يعتمدُ في تخزين وإدارة بياناتها على نظم قواعد البيانات عوضاً عن الملفات التقليدية التي كانت تعتمدُ عليها مواقع الإنترنت في تخزين بياناتها عند بداية ظهورها. وأصبح بالإمكان تطويرُ نماذج لصفحات الإنترنت يتمُّ استخدامها من قِبل متصفحات الإنترنت لكتابة الاستفسارات (Queries) ومن ثمَّ، تنفيذ هذه الاستفسارات على قاعدة بيانات الموقع. وبعد الحصول على نتيجة الاستفسارات تهيئ النتيجة باستخدام إحدى لغات المتصفحات، مثل: (Hyper Text Mark-up Language (HTML)) لعرضها من خلال المتصفح (Browser).

ومع دخول نظم إدارة قواعد البيانات في منظومة شبكة الإنترنت؛ فإن ذلك أعطاها زخماً جديداً من الأهمية، وضرورة البحث عن طرق وأساليب جديدة للاستخدام في تصميمها؛ بهدف تلبية الاحتياجات الجديدة التي تتطلبها تطبيقات الإنترنت، مثل الوسائط المتعددة من صورة وصوت.

الفصل الثاني

نمذجة بيانات المنظمة

يُعَدُّ «تجريد البيانات» (Data AbstraCtion) إحدى الخصائص الأساسية لقواعد البيانات؛ إذ يُمكِّن من إخفاء تفاصيل حِفظ البيانات عن المستخدِمين. ومن ثم؛ فإنه يعفيهم من الخوض في هذه التفاصيل عند تداولهم للبيانات المخزَّنة في قاعدة البيانات. ويُمكِّن نموذج البيانات - الذي يُعرَّف عادةً بأنه مجموعة من المفاهيم التي تمكِّن من وصف تركيبة (StruCture) مكونات قاعدة البيانات (Elmasri and Navathe, 2015) - من الوصول إلى هذا المستوى من التجريد. ويُقصَد بتركيبة قاعدة البيانات «نوعية البيانات» (Data Types)، والعلاقات فيما بينها، والقيود المفروضة عليها؛ ويجب أن تتحقق على أية حالة من الحالات التي قد تكون عليها قاعدة البيانات. كما توفر معظم نماذج البيانات بعض العمليات الأساسية لتداول البيانات من استرجاع لها وتحديث عليها.

> نموذج البيانات: هو مجموعة من المفاهيم التي تمكِّن من وَصْف تركيبة (StruCture) مكونات قاعدة البيانات.

ويُمكِن تصنيف نماذج البيانات وفق نوعية المفاهيم التي تستخدمها لوصف تركيبة قاعدة البيانات. فالنمذجة عالية المستوى أو المفاهيمية للبيانات (Modeling Data ConCeptual) توفر مفاهيم قريبة من إدراك المستفيدين للبيانات، في حين أن النمذجة متدنية المستوى أو المادية للبيانات (PhysiCal Data Modeling)؛ فتقدِّم مفاهيم لوصف تفاصيل تخزين البيانات على الحاسب الآلي، وهي موجهةٌ بشكلٍ عام إلى المتخصصين في الحاسب الآلي وليس إلى المستفيدين

منه. وبين هاتين النهايتين يُوجَد نوعٌ ثالث يُسمَّى النمذجة «التمثيلية» (Representational) أو «التطبيقية» (Implementation) للبيانات. وهذا النوع من نماذج البيانات يوفر مفاهيم يمكن فهمها من قِبل المستفيدين الذين لا يبعدون كثيراً عن الحاسب الآلي وطريقة تنظيم البيانات عليه. كما أن النمذجة التمثيلية للبيانات تخفي بعض تفاصيل تخزين البيانات، وفي الوقت نفسه يمكن استخدامها بشكلٍ مباشر على الحاسب الآلي.

تستخدم النمذجة المفاهيمية مفاهيم، مثل: الكينونة، والخاصية، والعلاقة. وتُمثل الكينونة شيئاً حقيقياً موجوداً على أرض الواقع أو مفهوماً معيناً. فمن الأشياء الموجودة على أرض الواقع الموظف، والطالب، والسيارة وما إلى ذلك من أشياء محسوسة. ومن أمثلة المفاهيم الحساب البنكي، والمشروع، والقسم الدراسي أو الإدارة، وما إلى ذلك من أشياء غير محسوسة؛ ولكنها ذات معنًى في بيئة المستفيد من قاعدة البيانات. أما الخاصية؛ فهي سمةٌ تصِف الشيء مثل: اسم الموظف أو مرتبته الوظيفية أو راتبه الشهري، في حين أن العلاقة ارتباط بين كينونتين أو أكثر. فعلى سبيل المثال: تُوجَد علاقةٌ بين الموظف والإدارة التي يعمل فيها، وتُسمَّى مثل هذه العلاقة علاقة «يعمل في». كذلك هو الحال بالنسبة لعلاقة الطالب بالقسم الدراسي أو الكلية الجامعية، وتُسمَّى «يدرس في». ويتطرق هذا الفصل لأحد النماذج المفاهيمية ويُسمَّى «نموذج كينونة - علاقة» (Entity-Relationship Model) الذي يُعَدُّ أكثر النماذج المفاهيمية عالية المستوى شيوعاً. أمَّا الفصل التالي؛ فيقدِّم مفاهيم إضافية تستخدم في النموذج المفاهيمى «كينونة - علاقة»، مثل: «التعميم» (Generalization)، و«التخصيص» (SpeCialization).

أمَّا النمذجة التمثيلية للبيانات أو التطبيقية؛ فهي أسلوب النمذجة المُستَخدَم في نظم إدارة قواعد البيانات على المستوى التجاري. ومن هذه النماذج «النموذج العلاقي» (Relational Model) الذي يُعَدُّ الأوسع انتشاراً في وقتنا الراهن، و«النموذج الشبكي» (Network Model) و«النموذج الهرمي» (HierarChiCal Model) اللذان كانا قيد الاستخدام وحتى وقت قريب.

أمَّا النمذجة المادية للبيانات؛ فهي نماذج تُستخدَم لوَصْف الكيفية التي يتمُّ فيها تخزين البيانات في ملفات على الحاسب الآلي من خلال توفيرها لطرق من شأنها تهيئة السجلات (أو تشكيلها) (ReCord Formatting) في الملفات، وترتيب السجلات داخل الملفات (ReCord Orderings)، والوصول إلى السجلات (ACCess Paths). أمَّا طرق الوصول إلى السجلات؛

فما هي إلا هياكل (StruCtures) من شأنها أن تعجِّل أو تسرِّع في عملية البحث والوصول إلى سجلات البيانات المخزنة في الملفات.

2-1 نمذجة البيانات وقواعد العمل باستخدام النمذجة المفاهيمية (ConCeptual Modeling):

لكلِّ منظمة من المنظمات - بغض النظر عن طبيعة عملها وحجمها - مجموعةٌ من الأُطُر التشريعية والتنظيمية التي تحكم عملها. ومن هذه الأطر: الأنظمة، واللوائح، والأوامر، والتعاميم، والاتفاقيات، والأعراف المرعية؛ على سبيل المثال فحسب. وتُحدِّد هذه الأطر في مجملها، عادةً، جميع وظائف المنظمة كما تحكم جميع إجراءاتها بشكلٍ تفصيلي. وتتيح هذه الأطر، في الكثير من الأحيان، مساحةً كافيةً للمسئولين في المنظمة لاتخاذ القرارات في حال كون الأطر المعمول بها لا تغطي جانباً معيناً من عملها بشكلٍ واضحٍ وصريح، أو الرجوع للجهة المختصة التي سنَّت الإطار الذي من المفترض العمل به إما لتفسيره وإيضاحه أو للتوجيه إزاء ما يتوجب القيام به.

وعند محاولة مكننة العمل في أية منظمة؛ يتوجَّب التقيد بجميع الأطر المعمول بها يدوياً ودونما استثناء. ونظراً لاستحالة مكننة جميع أعمال أية منظمة بشكلٍ كامل؛ فإنه يتم عادةً ترجمة الأطر المعمول بها إلى قواعد عمل (Business Rules) تتكون من جمل بسيطةٍ مرقمةٍ تخلو من الالتباس، تحصر وظائف المنظمة وإجراءاتها. ويتم تقسيم قواعد العمل هذه إلى ثلاث فئات، كما يلي:

1- قواعد عمل يتعذر مكننتها إما لعدم توفر التقنية اللازمة لمكننتها، أو لعدم جدوى مكننتها مالياً أو إدارياً. وقد يكون من ضِمن قواعد العمل هذه «وجوب حضور الموظف بالزِّي المناسب»، أو «الحضور والانصراف من العمل في وقتٍ مُحدَّد».

2- قواعد عمل يتوجب مكننتها؛ ولكنها لا تتعلق ببيانات المنظمة بشكلٍ مباشر، وإنما تتطلب كتابة إجراءات برمجية لتنفيذها. وتتمُّ مكننة مثل قواعد العمل هذه من خلال برامج تُكتَب خصيصاً لمثل هذه الإجراءات. وقد يكون من ضِمن قواعد العمل هذه أن «يُصرَف لكلِّ موظف سنوياً مكافأة مالية يُحدِّدها راتبه الشهري والدرجة التي يتحصل عليها في تقويم أدائه الوظيفي السنوي وفق

المعادلة التالية: (الراتب الشهري * 3 * (درجة الأداء الوظيفي÷ 4))، على أن تكون درجة الأداء الوظيفي كما يلي: (مقبول تعادل (1)، جيد تعادل (2)، جيدجداً تعادل (3)، ممتاز تعادل (4))».

3- قواعد عمل تتعلق ببيانات المنظمة بشكلٍ مباشر. ومن ضمن قواعد العمل التي تتعلق بالبيانات وأبسطها أسماء البيانات وتعاريفها. كما أن من ضِمن قواعد العمل هذه وتُعَدُّ أساسيةً في الغالبية العظمى من المنظمات، على سبيل المثال، تقييد البيانات، مثل: «ألَّا يمارس العمل في المنظمة مَنْ يزيد عمره على ستين عاماً»، أو أن يكون «لكلِّ موظف عنوان لبريده الإلكتروني».

وتقع مسؤولية حصر وتصنيف قواعد العمل كافةً في المنظمة على كاهل إداري البيانات أو مطوِّري النظم. وبعد عملية حصر وتصنيف قواعد العمل؛ يتمُّ تحديد تلك التي من الممكن مكننتها. وتتمُّ مناقشة قواعد العمل هذه ومراجعتها لأكثر من مرة مع المستفيدين (وهم الأشخاصُ المتعاملون معها). وبعد ذلك يتمُّ تحديدُ قواعد العمل التي تحتاج إلى برامج تتمُّ كتابتها للتمكُّن من مكننتها، وتلك التي تتعلق مباشرةً ببيانات المنظمة. ويُعَدُّ النوعُ الأخير من قواعد العمل؛ هو النوع الذي تتعاطى معه نظم إدارة قواعد البيانات، وهو النوع الذي سيتمُّ التطرُّق له والتعامل معه في هذا الكتاب. أمَّا النوع الثاني الذي يمكِن مكننته من قواعد العمل؛ فهو محل اهتمام تطوير النظم المعلوماتية بجوانبها البرمجية والذي عادةً ما تتم تغطيته من خلال لغات البرمجة، ويخرج عن إطار هذا الكتاب.

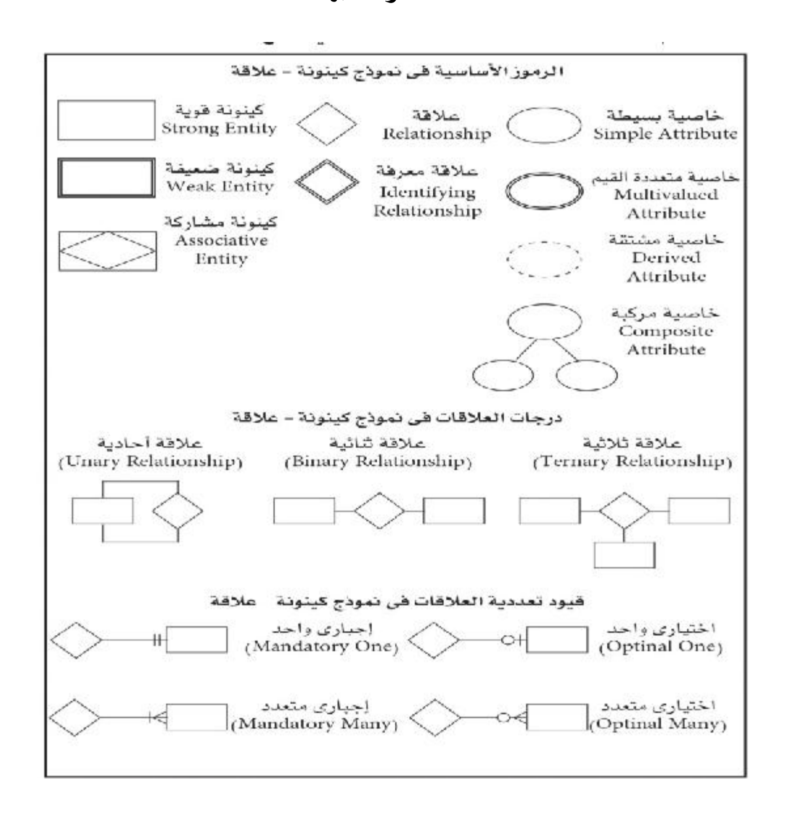
وتُعَدُّ النمذجة المفاهيمية لبيانات المنظمة؛ الخطوة الأولى لتطوير قاعدة بيانات أية منظمة، كما يُعَدُّ نموذج البيانات «كينونة - علاقة» (Model Entity-Relationship) أكثر نماذج البيانات المفاهيمية شيوعاً. ويُعزَى سبب شيوع هذا النموذج لعدة عوامل من ضِمنها السهولة النسبية في الاستخدام، وإمكانية نمذجة البيانات وفق هذا النموذج باستخدام ما توفره غالبية «أدوات هندسة البرمجيات» (CASE Tools) من أدوات مخصصة للنمذجة، هذا بالإضافة إلى الاعتقاد السائد بأن الكينونات والعلاقات هي مفاهيم نمذجة لها القدرة على تمثيل الأشياء بشكلٍ أقرب ما يكون من وجودها في الطبيعة.

ونموذج البيانات «كينونة - علاقة»، في كثير من الأحيان، هو أداة للتواصل بين الفنيين من مصمِّمي قواعد البيانات والمستفيدين النهائيين من النظم؛ وذلك خلال فترة تحليل النظام. كما

يُستخدَم نموذج «كينونة - علاقة» لبناء نموذج بيانات مفاهيمى يمثل تركيبة قاعدة البيانات والقيود عليها بمعزل عن البرمجيات التي ستُستخدَم لبناء قاعدة البيانات، متضمناً ذلك نظام إدارة قاعدة البيانات ونموذج البيانات التمثيلي أو التنفيذي الذي سيُستخدَم لبنائها. وتُعَدُّ النمذجة المفاهيمية لقاعدة البيانات مرحلةً ذات أهمية كبيرة في تطوير «نظم تطبيقية» (AppliCation Programs) ناجحة. لذلك؛ فإن مصمِّمي قواعد البيانات يأخذون الوقت الكافي في هذه المرحلة؛ لمناقشة النموذج المفاهيمى مع المستفيدين للتأكد من أنه يعكس كافة بياناتهم والقيود عليها؛ وذلك قبل الانتقال إلى المرحلة التالية.

2–1–1 مكوّنات نموذج البيانات «كينونة – علاقة»:

نموذج البيانات «كينونة – علاقة»؛ هو تمثيل منطقي مُفصَّل للبيانات الموجودة في المنشأة أو «منطقة العمل» (Business Area). ويمثل نموذج البيانات «كينونة – علاقة» من خلال مجموعةٍ من الكينونات، وخصائصها، والعلاقات فيما بينها. ويوضِّح النموذج من خلال مخطط «كينونة – علاقة» الذي يتمُّ تمثيله من خلال مجموعة من الرسومات. ويبيِّن الشكل رقم (2–1) أشكال الرموز الأساسية المُستخدَمة في نموذج «كينونة – علاقة» ومعانيها.



شكل رقم (2–1): الرموز الأساسية المستخدمة في نموذج كينونة – علاقة ومعانيها.

ولشرح مكونات نموذج البيانات «كينونة - علاقة» سنستخدم مثالاً لأحد تطبيقات قواعد البيانات المستمدَّة من نظام لتسجيل الطلبة في إحدى الجامعات الأهلية. وسنفترض هنا أنه بعد عملية جمع متطلبات النظام وتحليلها في أثناء مرحلة تحليل النظام؛ تم استخلاص قواعد العمل التالية من قِبل مُصمِّمي قواعد البيانات:

قواعد العمل المعمول بها في الجامعة:

تنفذ الجامعة الأهلية مجموعةً من المواد الدراسية في كلِّ فصل دراسي. ومن قواعد العمل المتبعة في الجامعة الأهلية، ما يلي:

1- يُوجَد في الجامعة عددٌ من الأقسام الدراسية، ولكلِّ قسم (Department) من الأقسام العلمية رمز (Department_ID) يميّزه عن بقية الأقسام، واسم (Name).

2- يعمل في الجامعة عددٌ من أعضاء هيئة التدريس، ولكل عضو هيئة تدريس (FaCulty)[1] رمز (FaCulty_ID) يميّزه عن بقية أعضاء هيئة التدريس، واسم (Name) يتكون من (الاسم الأول (FName) واسم العائلة (LName))، وراتب شهري (Salary)، وتاريخ ميلاد (Date Of Birth (DOB)) ورقم هاتف (Phone_No).

3- يدرس في الجامعة عددٌ من الطلاب، ولكلِّ طالب (Student) رمز (Student_ID) يميّزه عن بقية الطلاب في الجامعة، واسم (Name) يتكون من (الاسم الأول (FName) واسم العائلة (LName))، وعنوان بريدي (Address) يتكون من (اسم الشارع (Street)، واسم المدينة (City)، والرمز البريدي (Zip_Code)).

4- تنفذ الجامعة مجموعةً من المواد الدراسية، ولكل مادة دراسية (Course) رمز (Course_ID) يميّزها عن بقية المواد الدراسية التي تنفذها الجامعة، واسم (Title)، وعدد وحدات (أو ساعات) دراسية (Units).

5- تُنفَّذ (أو تُعقَد) كلُّ مادة دراسية من خلال مجموعة (أو شعبة) دراسية (SeCtion) واحدة أو أكثر في الفصل الدراسي الواحد، أو قد لا تُنفَّذ (أو تُعقَد) أية مجموعة (أو شعبة) للمادة

الدراسية في فصل دراسي معين، ولكل مجموعة دراسية رمز (SeCtion_ID) يتكون من (رقم المجموعة، والفصل الدراسي المنفذة فيه (Semester)، والسنة الدراسية المنفذة فيها (Year)). أما رقم المجموعة (SeCtion_No) فهو رقم (مثل: 1، 2، 3، ... إلخ) يميّز المجموعة عن بقية المجموعات المنفِّذة للمادة الدراسية نفسها (وفي نفس الفصل والسنة الدراسيين)؛ ولكنه لا يميزها بشكل منفرد عن بـقية المجموعات الدراسية المنفِّذة للمواد الدراسية الأخرى في الجامعة. كما أن لكلِّ مجموعة دراسية موقعاً من مواقع الجامعة لتنفيذها فيه (LoCation).

6- قد يكون للمادة الدراسية الواحدة مجموعةٌ من المتطلبات الدراسية، أو قد لا يكون للمادة الدراسية أية متطلبـات دراسية. كما أن المادة الدراسية الواحدة قد تكون متطلباً لأكثر من مادة دراسية أو قد لا تكون متطلباً لأية مادة دراسية.

7- يعمل في (works for) كل قسم من أقسام الجامعة عضو هيئة تدريس واحد أو أكثر، وكل عضو من أعضاء هيئة التدريس يعمل في قسم دراسي واحد فقط.

8- كلُّ عضو هيئة تدريس في الجامعة مؤهل (Qualified) لتدريس مادة دراسية واحدة على الأقل، وقد يتوفر للمادة الدراسية الواحدة أكثر من عضو هيئة تدريس مؤهلٍ لتدريسها أو قد لا يُوجَد من أعضاء هيئة التدريس من هو مؤهل لتدريس المادة.

9- عندما يتأهل عضو هيئة التدريس لتدريس مادةٍ ما لأول مرة؛ يكون هنالك تاريخٌ لتأهيله (QualifiCation Date) يُحدِّد تاريخ تأهل عضو هيئة التدريس لتدريس المادة الدراسية.

10- تُدار (Administered) كلُّ مادة دراسية من قِبل قسم دراسي واحد من أقسام الجامعة، ويدير كلُّ قسم مادةً دراسيةً واحدةً على الأقل.

11- قد يُسجّل (Enrolls) الطالب الواحد في أكثر من مجموعة (أو شعبة) دراسية، أو قد لا يُسجّل في أية مجموعة (أو شعبة) دراسية، والمجموعة (أو الشعبة) الدراسية الواحدة قد لا يُسجَّل فيها أيُّ طالب أو قد يُسجّل فيها أكثر من طالب.

12- عندما يُسجَّل طالب في مجموعة دراسية تكون له درجة (Grade) تُعطَى عند انتهائه من الدراسة في المجموعة.

13- يتخصص كلٌّ طالب (Majors) في قسمٍ دراسيٍّ واحدٍ فقط، ويتخصص في القسم الدراسي الواحد أكثرُ من طالب.

14- يُكلَّف (Assigned) كلُّ عضو هيئة تدريس بالتدريس لمجموعة (أو شعبة) دراسية واحدة أو أكثر، وقد لا يُكلَّف عضو هيئة التدريس بأيَّة مجموعة (أو شعبة) دراسية، والمجموعة (أو الشعبة) الدراسية الواحدة تُكلَّف لعضو هيئة تدريسٍ واحدٍ فقط.

2-1-2 المكوِّنات الأساسية لنموذج البيانات «كينونة - علاقة»:

يتكوَّن نموذج البيانات المفاهيمي «كينونة - علاقة» من ثلاثة مكونات رئيسية، هي:

- الكينونة.

- الخاصية.

- العلاقة.

وفيما يلي شرحٌ لهذه المكونات الرئيسية.

2–1–2–1 الكينونة (**Entity**):

الكينونة: هي شخصٌ أو مكانٌ أو شيءٌ أو حدثٌ أو مفهومٌ في بيئة المنظمة، ويُراد الاحتفاظ ببيانات عنها. ومن أمثلة كينونات الأشخاص؛ كينونة الموظف وكينونة الطالب وكينونة العميل. أمَّا كينونة المدينة، وكينونة السوق، وكينونة المبنى؛ فتُعَدُّ أمثلةً لكينونات الأماكن. ومن أمثلة كينونات الأشياء؛ كينونة سيارة، وكينونة منتج، وكينونة جهاز. أمَّا كينونات المفاهيم؛ فمن أمثلتها كينونة مادة دراسية، وكينونة رحلة جوية، وكينونة حساب بنكي. ويعني هذا أنه ليس بالضرورة أن تمثل الكينونات أشياءً ملموسةً لها وجودها الفيزيائي في الطبيعة، ولكنها قد تمثل أشياءً أخرى لها مفهومها في بيئة المنظمة.

الكينونة: هي شخصٌ أو مكانٌ أو شيءٌ أو حدثٌ أو مفهومٌ في بيئة المنظمة ويُراد الاحتفاظ ببيانات عنها.

2-1-2-1-1 الفرق بين فئة الكينونة، وحالة من حالات الكينونة (Entity Type Versus Entity InstanCe):

فئة الكينونة: هي تمثيلٌ لمجموعةٍ من الحالات للأشياء التي لها خواص مشتركة. ويتم التفريق عادةً بين فئة (أو نوع) الكينونة (Entity Type or Entity Class) عن حالات الكينونة (Entity InstanCes). فكينونة الطالب في الواقع؛ هي نوعٌ أو فئةٌ لمجموعة من حالات الطلبة، فالطالب أحمد يختلف باعتباره حالة عن الطالب صالح وعن الطالب عبدالعزيز؛ غير أن جميعهم يشتركون في نفس الخواص؛ فكلٌّ منهم لديه اسمٌ أول، واسمٌ لعائلته، ورقمٌ خاص به (كالرقم الجامعي للطالب على سبيل المثال)، كما أن كلاً منهم لديه هاتف وعنوان بريدي.

فئة الكينونة: هي تمثيلٌ لمجموعةٍ من الحالات للأشياء التي لها خواص مشتركة.

ويتمُّ تمثيل الحالات للكينونات في نموذج «كينونة - علاقة»؛ من خلال تمثيل الفئة (أو النوع) التي تتبعها هذه الحالات دون تمثيل كلِّ حالة بشكلٍ منفرد في النموذج؛ فمثلاً يتمُّ تمثيل جميع الطلاب من خلال فئة الكينونة التي يتبعونها، وهي كينونة طالب في نموذج البيانات «كينونة - علاقة» عوضاً عن إدراجهم جميعاً في النموذج. وبهذه الطريقة يمكن تمثيلُ بيانات أية منظمة من خلال تمثيل فئات الكينونات التي تحتويها. كما أن هذه الطريقة تُعَدُّ مختصرةً جداً؛ مما يساعد على التعرُّف على بيانات المنشأة وتمثيلها بشكلٍ مُبسَّط.

وتُمثَّل الكينونات في نموذج «كينونة - علاقة» على شكل مستطيل يُكتب بداخله اسم فئة (أو نوع) الكينونة. ومن الإرشادات التي تتبع عادةً عند تسمية الكينونات ما يلي (Hoffer et al, 2018):

1- اسم فئة الكينونة «مُسمًّى فردي» (Singular Noun) مثل: «عميل» (CUSTOMER)، وطالب (STUDENT). إلا أنَّ مُسمًّى فئة الكينونة قد يكون «مُسمًّى جماعياً» (Noun Plural)، مثل: (CUSTOMERS)، و(ACCOUNTS)، ويكون هذا في الحالة التي تكون فيها قراءة المخطط أفضل من استخدام مُسمًّى فردي للكينونة.

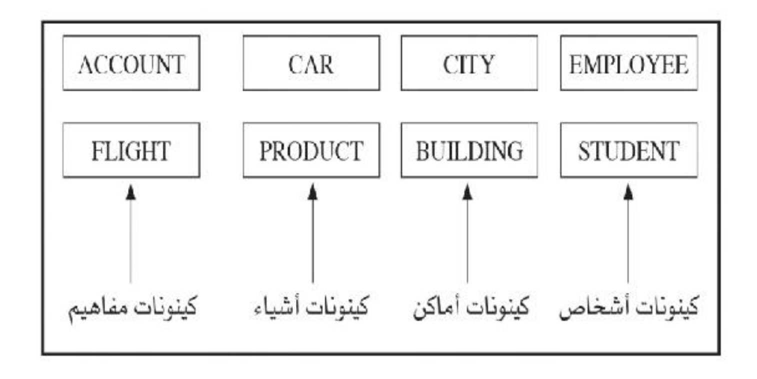
2- يجب أن يكون مُسمَّى الكينونة مخصصاً للمنظمة؛ فعلى سبيل المثال: قد يُستخدَم المُسمَّى «طالب» (STUDENT) في إحدى الجامعات أو المدارس، ويُستخدَم المُسمَّى «متدرب» (TRAINEE) في المنظمات التي تقوم بالتدريب التطبيقي (أو العملي) عوضاً عن التدريس النظري (أو الأكاديمي). كما يجب أن يكون المسمَّى ذا طبيعةٍ وصفيةٍ تصف المقصود من الكينونة بشكلٍ مختلفٍ عن بقية الكينونات في المنظمة.

3- مُسمَّى فئة الكينونة يجب أن يكون محدداً بأقل قدرٍ ممكن من الكلمات. فمثلاً يُستخدَم الاسم «تسجيل» (ENROLLMENT) لتمثيل فئة كينونة تسجيل الطلبة في المواد الدراسية عوضاً عن استخدام «التسجيل في مادة دراسية» (ENROLLMENT IN A CLASS). وعادةً ما يُفهَم معنى المسمَّى من خلال علاقة فئة الكينونة بالفئات الأخرى في المخطط.

4- في حالة تمثيل الفئة لحدثٍ معين؛ فإن مُسمَّى الفئة يكون ممثلاً لنتيجة الحدث. فعلى سبيل المثال عند تسمية حدث تسجيل الطالب في مادة معينة؛ يكون مُسمَّى الفئة (ENROLLMENT)، وهو ممثل لنتيجة الحدث وهي عملية التسجيل.

5- عند استخدام اسم معين لفئة كينونة؛ فإنه يجب استخدام نفس المُسمَّى في كافة مخططات «كينونة - علاقة» الخاصة بالمنظمة.

ويوضِّح الشكل رقم (2–2) بعض الأمثلة لفئات الكينونات وطريقة تمثيلها في مخطط كينونة علاقة.



شكل رقم (2–2): أمثلة لفئات الكينونات وطريقة تمثيلها في مخطط «كينونة – علاقة».

2–1–2–1–2 خصائص الكينونات (Entity Attributes):

ترتبط كلُّ فئة كينونة بعددٍ من الخصائص (Attributes). والخاصية هي صفة أو سمة لفئة الكينونة التي يُراد تمثيل بياناتها في قاعدة البيانات الخاصة بالمنظمة. فالاسم الأول والاسم الأوسط واسم العائلة قد تكون بعض خصائص كينونة عميل (CUSTOMER)، ونوع السيارة ورقم لوحتها والبلد الذي صُنعت فيها قد تكون بعض خصائص كينونة مركبة قيادة (VEHICLE). ويمكن تصنيف خصائص الكينونات إلى أربعة أنواع رئيسية، وهي:

1- الخاصية البسيطة (Simple Attribute): هي الخاصية التي لا يمكن أن تنقسم إلى خصائص فرعية، وبحيث إنها لا يمكن أن تأخذ أكثر من قيمة واحدة فقط، مثل: الاسم الأول للموظف Employee_First_Name))، أو نوع المركبة (VehiCle_Type)، أو رقم الموظف (Employee_Number). ويُرمز للخاصية البسيطة بالشكل البيضاوي مدوناً بداخله اسم الخاصية.

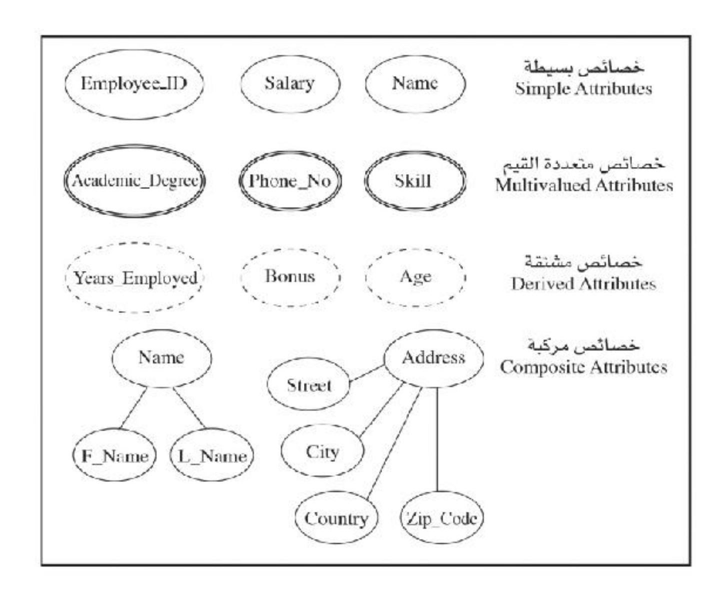
2- الخاصية المركبة (Composite Attribute): هي الخاصية التي تتكوَّن من مجموعةٍ من الخصائص البسيطة، مثل الاسم؛ إذ إنه قد يتكون من الاسم الأول، واسم الأب، واسم العائلة. كذلك هو الحال بالنسبة للعنوان البريدي الذي قد يتكون من: اسم الشارع (Street)، واسم المدينة (City)، واسم البلد (Country)، ويرمز للخاصية المركبة بالشكل البيضاوي أيضاً، ويُكتب بداخلها اسم الخاصية، وبحيث يتفرع منها الخصائص البسيطة المكونة لها.

3- الخاصية المتعددة القيم (Multivalued Attribute): هي الخاصية التي من الممكن أن تأخذ أكثر من قيمةٍ، مثل خاصية المهارة (Skill) للموظفين. فقد يكون للموظف، على سبيل المثال، المهارة في البرمجة بأكثر من لغة برمجة، أو قد يكون من المهم للمنظمة تدوين مهارات (أو قدرات) الموظفين؛ من حيث اللغات التي يمكن التخاطب بها. كذلك هو الحال بالنسبة لأرقام الهواتف والدرجات العلمية عندما تتعَدَّد عند الموظفين، على سبيل المثال، وترغب المنظمة في تمثيل بياناتها ضمن قاعدة البيانات. ويرمز للخاصية متعددة القيم بالشكل البيضاوي المزدوج الخطوط يدون بداخله اسم الخاصية.

4- الخاصية المشتقة (Derived Attribute): هي الخاصية التي يمكن استنتاجها من خلال خاصية (أو خصائص) أخرى للكينونة، مثل العمر (Age) الذي يمكن حسابه بعملية طرح

تاريخ اليوم (الذي يوفره نظام الحاسب الآلي) من خاصية تاريخ الميلاد التي تكون مصاحبة لفئة الكينونة. كذلك هو الحال بالنسبة لتاريخ تقاعد الموظف (Retirement_Date) الذي يُعَدُّ خاصيةً مشتقةً يمكن حسابها من خلال عملية جمع خاصية تاريخ ميلاد الموظف مع السِّن التقاعدية المسموح بها في أنظمة المنظمة أو الدولة. وكذلك هو الحال بالنسبة لخاصية المكافأة السنوية (Bonus) للموظف، في بعض المنظمات، التي يمكن حسابها كنسبة من خاصية راتب الموظف (Salary). ويُرمَز للخاصية المشتقة في نموذج «كينونة - علاقة» بالشكل البيضاوي المنقط يُكتب بداخله اسم الخاصية.

وعند تسمية خصائص فئات الكينونات، على اختلاف أنواعها؛ يكون الحرف الأول من الخاصية حرفاً كبيراً (Capital Letter). وفي حالة كون اسم الخاصية مركباً؛ فإنه يتم الربط ما بين الكلمات المكونة لاسم الخاصية بالعلامة "_". كذلك يمكن استخدام الاختصارات، ولكنها يجب أن تكون مفهومةً ومدونةً بشكلٍ واضحٍ ضمن وثائق النظام. ويحتوي الشكل رقم (2-3) على بعض الأمثلة التي توضح طرق تمثيل الأنواع الأربعة من الخصائص.

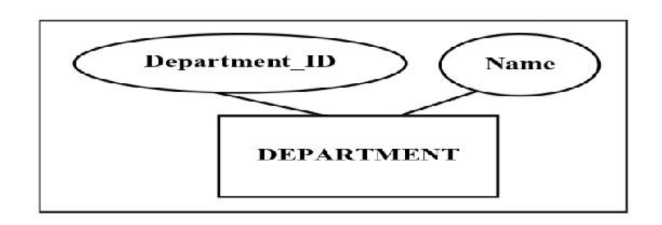


شكل رقم (2–3): أمثلة توضح طرق تمثيل الأنواع الأربعة من الخصائص في نموذج «كينونة – علاقة».

تُربط كلُّ خاصية بفئة الكينونة التابعة لها بخط مستقيم. فعلى سبيل المثال، لنأخذ قاعدة العمل الأولى والثانية في الجامعة الأهلية، وقد سبق ذكرهما أعلاه في هذا الفصل، ونحاول تمثيلهما في مخطط «كينونة - علاقة».

- **قاعدة العمل (1):** يُوجَد في الجامعة عددٌ من الأقسام الدراسية، ولكلِّ قسم (Department) من الأقسام العلمية رمز (Department_ID) يميِّزه عن بقية الأقسام، واسم (Name).

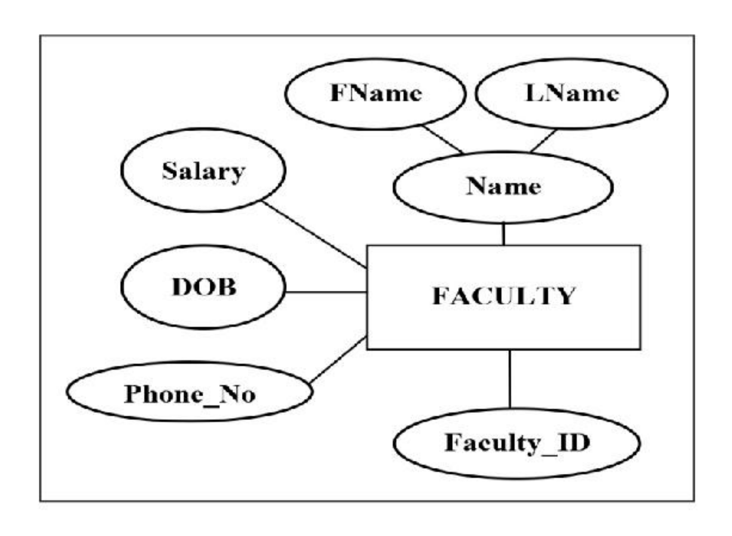
نظراً لوجود عدة حالات للقسم الدراسي فمنها، على سبيل المثال، قسم الحاسب الآلي وقسم الرياضيات... إلخ، وهو ما نصَّت عليه قاعدة العمل؛ تُمثَّل قاعدة العمل هذه في مخطط «كينونة - علاقة» كفئة كينونة بمُسمَّى (DEPARTMENT)، ولها خاصيتان بسيطتان هما رمز القسم (Department_ID) واسم القسم (Name) كما هو موضح في الشكل رقم (2-4).



شكل رقم (2–4): تمثيل القسم الدراسي كفئة كينونة ذات خاصيتين بسيطتين.

- **قاعدة العمل (2):** يعمل في الجامعة عددٌ من أعضاء هيئة التدريس، ولكلِّ عضو هيئة تدريس (FaCulty) رمز (FaCulty_ID) يميزه عن بقية أعضاء هيئة التدريس، واسم (Name) يتكون من (الاسم الأول (FName) واسم العائلة (LName))، وراتب شهري (Salary)، وتاريخ ميلاد (Date Of Birth (DOB))، ورقم هاتف (Phone_No).

نظراً لوجود العديد من حالات أعضاء هيئة التدريس في الجامعة، تمثل قاعدة العمل هذه في مخطط «كينونة – علاقة» كفئة كينونة بمُسمَّى (FACULTY)، ولها أربع خصائص بسيطة، هي: رمز عضو هيئة التدريس (FaCulty_ID) وتاريخ ميلاده (DOB)، وراتبه الشهري (Salary)، ورقم هاتفه (Phone_No)، وخاصية مركبة هي اسمه (Name) التي تتفرع إلى خاصيتين بسيطتين، هما: الاسم الأول (FName) واسم العائلة (LName)؛ وذلك كما هو مُوضَّح في الشكل رقم (2–5).



شكل رقم (2-5): تمثيل عضو هيئة التدريس كفئة كينونة ذات أربع خصائص بسيطة وخاصية مركبة.

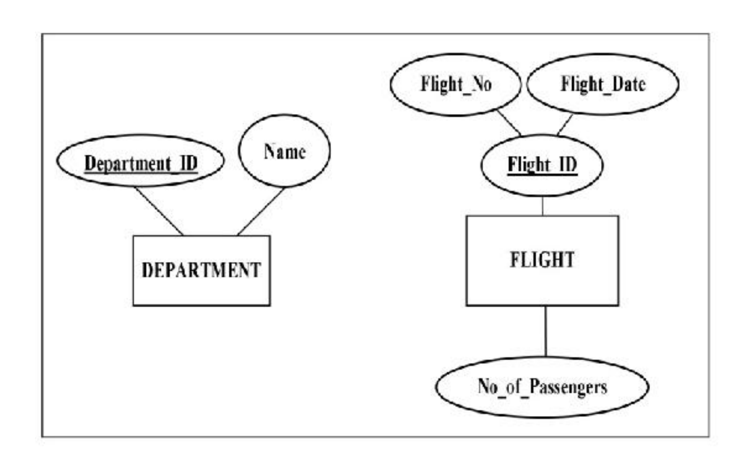
2-1-2-1-3 الخاصية المميزة (أو المعرفة) (Identifying Attribute) لفئة الكينونة:

الخاصية المميّزة لفئة الكينونة؛ هي واحدةٌ أو أكثر من خصائص فئة الكينونة؛ بحيث تحدِّد هذه الخاصية بشكلٍ منفردٍ كلَّ حالة من حالات الكينونة، وفي الوقت نفسه لا تتغير بـتغير الزمن. وبمعنى آخر؛ تُستخدَم الخاصية المميزة للتفريق ما بـين الحالات التي تمثلها فئة الكينونة. فعلى سبيل المثال: الخاصية المميزة لكينونة القسم الدراسي في الجامعة الأهلية هي رمز القسم؛ وذلك لأن رمز القسم يختلف باختلاف الأقسام العلمية ودون تكرار؛ بـحيث لا يمكن أن يكون لقسمين دراسيين مختلفين نفس الرمز. كما أن رمز القسم لا يتغير بـتغير الزمن. أما بالنسبة لفئة كينونة أعضاء هيئة التدريس (FACULTY)؛ فإن الخاصية المميزة لفئة الكينونة هي رمز عضو هيئة التدريس (FaCulty_ID)؛ لأنه يميز كلَّ عضو هيئة تدريس عن بقية أعضاء هيئة التدريس ولا يتغير بـتغير الزمن. أما بالنسبة لبـقية الخصائص في كينونة أعضاء هيئة التدريس، مثل: تاريخ الميلاد ورقم الهاتف؛ فإنها لا تصلح لأن تكون مميزاً لفئة الكينونة. فتاريخ الميلاد قد يتكرر بين الموظفين، ومن ثم لا يمكن استخدامه للتفريق بين حالات الكينونة. أما رقم الهاتف، لو افترضنا أنه لا يتكرر بمعنى عدم اشتراك أكثر من عضو هيئة تدريس لنفس رقم الهاتف؛ فإنه لا يصلح أيضاً لأن يكون خاصيةً مميزةً؛ لأنه يتغير بتغيُّر الزمن، وذلك عند انتقال عضو هيئة التدريس من قسم إلى قسم أو مكتب آخر داخل الجامعة.

في بعض الأحيان قد لا يُوجَد لفئة كينونة ما خاصية واحدة تصلح أن تكون خاصية مميزة. في هذه الحالة؛ يتم اختيارُ أكثر من خاصية بحيث تمثل في مجملها خاصيةً مميزةً للكينونة بإمكانها التفريق بين حالات فئة الكينونة بشكلٍ منفرد. فعلى سبيل المثال: وبافتراض عدم وجود خاصية رمز عضو هيئة التدريس، وأن أسماء أعضاء هيئة التدريس لا تتكرر؛ يمكن استخدام اسم عضو هيئة التدريس الذي يتكون من الاسم الأول واسم العائلة كخاصية مميزة لفئة «كينونة أعضاء هيئة التدريس» (FACULTY). وفي هذه الحالة تصبح الخاصية المميزة خاصيةً مميزةً مركبةً (Composite Identifying Attribute). ولكن لو افترضنا أن أسماء أعضاء هيئة التدريس تتكرَّر؛ فإنه بالإمكان استخدام اسم عضو هيئة التدريس وتاريخ ميلاده كخاصية مميزة مركبة عوضاً عن الخاصية المركبة المتمثلة في اسم عضو هيئة التدريس فقط.

وكمثالٍ آخر للخصائص المميزة المركبة؛ لنفترض وجود الكينونة رحلة طيران (FLIGHT) التي لديها الخاصية رمز الرحلة (Flight_ID) الذي يتكوَّن من رقم الرحلة (Flight_No) وتاريخ الرحلة (Flight_Date)، والخاصية عدد الركاب (Of_Passengers_No). يتم اختيار الخاصية رمز الرحلة المكونة من رقم الرحلة وتاريخ الرحلة كخاصية مميزة؛ إذ إنه لا يمكن استخدام أيٍّ من الخاصية رقم الرحلة أو تاريخ الرحلة كلٌّ على حِدَة كمميز. فرقم الرحلة يتكرَّر ولكن بتواريخ مختلفة. ومن ثم؛ فإنه لا يميز كلَّ رحلة على حِدة بشكلٍ منفرد. كذلك هو الحال بالنسبة لتاريخ الرحلة؛ لأنه من الممكن أن تكون هنالك أكثر من رحلة في التاريخ نفسه، ومن ثم لا تصلح هذه الخاصية أيضاً لتصبح مميزاً لفئة الكينونة؛ لأنها قد تتكرر لأكثر من رحلة.

وللتفريق بين الخاصية المميَّزة وبقية خصائص فئة الكينونة في مخطط «كينونة - علاقة»، يُكتَب اسم الخاصية المميزة وتحتها خط (Underlined) كما هو مُوضَّح في الشكل رقم (2-6) الذي يحتوي على مثالين: أحدهما لفئة كينونة الرحلة ذات الخاصية المميزة المركبة؛ وثانيهما لفئة كينونة القسم الدراسي ذات الخاصية المميزة البسيطة.



شكل رقم (2–6): التفريق بين تمثيل الخاصية المميزة، وبقية خصائص الكينونة في مخطط «كينونة – علاقة».

قد يُوجَد من ضِمن خصائص فئة كينونة معينة أكثر من خاصية تصلح لأن تميِّز بين حالات الكينونة. فعلى سبيل المثال: قد يكون لفئة كينونة الموظفين (EMPLOYEE) في إحدى المنظمات خاصية تحتوي على رمز الموظف (Employee_ID)، وخاصية أخرى تحتوي على رقم السجل المدني للموظف (SoCial_IdentifiCation_No). في هذه الحالة كلتا الخاصيتين تصلحان لأن تكونا خاصية مميزة لفئة الكينونة. وفي هذه الحالة يُطلق على كل خاصية من الخاصيتين مُسمَّى خاصية مميزة مرشحة (Candidate Identifying Attribute)؛ بمعنى أنه بالإمكان استخدام أيٍّ منهما مميزاً لفئة الكينونة. ولمصمِّمي قاعدة البيانات الحرية في اختيار إحدى الخصائص المميزة المرشحة لفئة الكينونة لتصبح الخاصية المميزة لفئة الكينونة. وتتم عملية الاختيار عند وجود أكثر من خاصية مرشحة وفق بعض المعايير التي بإمكان المصمِّمين اتباعها، ومنها ما يلي (BruCe, 1992):

1- ألَّا تتغير قيمة المميِّز لكلِّ حالة من حالات فئة الكينونة بتغير الزمن. فعلى سبيل المثال لا يُحبذ استخدام الاسم الثلاثي حتى وإن ميَّز بين الموظفين في المنظمة بشكلٍ منفرد، في بعض الدول؛ لأنه بإمكان الموظف أن يغيِّر اسمه بشكلٍ رسمي عند رغبته في ذلك. ويعني هذا أن المميز لهذا الموظف يجب أن يتغيَّر من خلال تحديثه في قاعدة البيانات عندما تحدث مثل هذه الحالة.

2- أن تكون لكلِّ حالة من حالات فئة الكينونة قيمةٌ، وأن هذه القيمة لا يمكن أن تكون غائبة (Null). وفي حالة اختيار مميز مركب مثل (Flight_ID)؛ فإنه يجب التأكد من أن كافة

الخصائص المكوِّنة للمميز سيكون لها قيمٌ، وأن قيم كلٍّ من هذه الخصائص لا يمكن أن تكون غائبة.

3- أن يكون المميِّز لفئة الكينونة هو المميِّز المرشح الأقل عدداً من الحقول.

وعند ارتباط كينونةٍ ما بخاصية، وأن قيمة هذه الخاصية لا يمكن أن تكون غائبةً في أية لحظة من اللحظات، وفقاً لقواعد العمل المعمول بها؛ فإنه يتمُّ توضيح ذلك على مخطط «كينونة - علاقة» من خلال وضع خط متعرج تحت اسم الخاصية. وبذلك يكون هناك اختلافٌ بين مثل هذه الخاصية والخاصية المميِّزة للكينونة. أما إذا كانت الخاصية فريدة، أي لا يمكن أن تتكرَّر قيمتها بين حالات الكينونة؛ ولكنها قد تكون غائبةً من بعض حالات الكينونة؛ فيتمُّ توضيح ذلك على المخطط؛ من خلال وضع خط متقطع تحت مُسمَّاها. وفي حال كانت الخاصية فريدةً وفي نفس الوقت لا يمكن أن تكون قيمتها غائبة؛ فيتمُّ توضيح ذلك من خلال وضع كلا النوعين من الخطوط تحتها (المتعرج والمتقطع)؛ للدلالة على ذلك. وتُعَدُّ خاصية رقم السجل المدني (SoCial_IdentifiCation_No)، عندما تكون هناك خاصية أخرى مميزة لكينونة ما غير رقم السجل المدني، مثالاً لهذا النوع الأخير من الخصائص؛ لكونها لا تتكرر بين حالات الكينونة كما أنه يتوجَّب وجودها لكل حالة من حالاتها.

2–1–2–1–4 قواعد تسمية الخصائص:

هنالك بعض القواعد التي تُتبع عادةً عند تسمية الخصائص؛ إضافةً إلى القواعد الرئيسية التي تُتبع لتسمية الكينونات، ومن هذه القواعد، ما يلي:

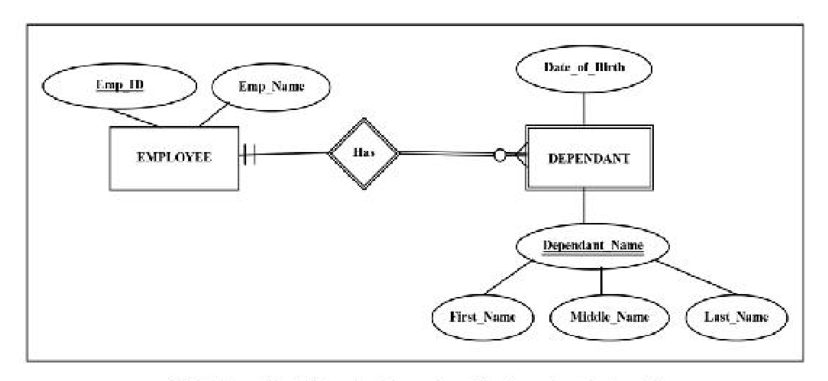
1- يكون اسم الخاصية اسماً، مثل: رقم الطالب (Student_Number)، والتخصص (Major)، وتاريخ الميلاد (Date_Of_Birth)... إلخ. ولكون الخاصية مفهوماً (مثل رقم الرحلة (Flight_Number)) أو سِمة فيزيائية (مثل الوزن (Weight)) للشيء قيد التمثيل؛ فإنه من الطبيعي أن توصف بأسماء لتمثيلها في مخطط «كينونة - علاقة».

2- يجب أن يكون اسم الخاصية فريداً (Unique) لا يتكرَّر ضِمن أسماء خصائص فئة الكينونة، ويُحبذ أن يكون فريداً أيضاً ضِمن خصائص جميع الكينونات في المخطط.

2–1–2–1–5 الكينونة الضعيفة (Weak Entity):

إن غالبية فئات الكينونات التي يتمُّ التعرُّف عليها في أيَّة منظمة؛ هي فئات كينونات قوية (Strong Entities). وجميع الكينونات التي تطرَّقنا إليها حتى الآن هي من هذا النوع من فئات الكينونات. فالكينونة القوية تُوجَد مستقلةً عن بقية الكينونات ولها خاصيتها المميزة التي تتمايز بها بين ما تمثله من حالات.

على النقيض من ذلك؛ فإنَّ الكينونة الضعيفة لا تُوجَد مستقلةً؛ بل تعتمد في وجودها على فئة كينونة أخرى بدونها تصبح الكينونة الضعيفة غير ذات معنى في مخطط «كينونة - علاقة» الذي يمثل قواعد عمل المنظمة. وتُسمَّى الكينونة التي تعتمد عليها الكينونة الضعيفة في وجودها «بالكينونة المُعرفة» أو «الكينونة المالكة». وعلى خلاف الكينونة القوية؛ فإنه لا يُوجَد للكينونة الضعيفة خاصية مُميزة تمايز بين حالاتها بشكلٍ منفرد؛ بل يُوجَد فيها خاصيةٌ مميزةٌ جزئيةٌ تمايز بين بعض حالاتها. ويحتوي مخطط «كينونة - علاقة» الموضَّح في الشكل رقم (2-7) على مثال لكينونة ضعيفة تمثل الأشخاص الذين يعولهم الموظف (DEPENDANT)، في منظمةٍ ما، والكينونة المالكة وهي كينونة الموظف (EMPLOYEE). وتُعَدُّ كينونة الأشخاص الذين يعولهم الموظف كينونةً ضعيفةً؛ لأن هذه الكينونة لا وجود لها وغير ذات معنى بدون وجود كينونة الموظف. فلو أخذنا أيَّة حالة من حالات الكينونة الضعيفة، ولنقل محمد صالح عبدالله؛ فإن هذه الحالة لا وجود لها؛ وإن وُجِدَت فإنها غير ذات معنى ما لم يُوجَد صالح عبدالله (وهو والد أو معيل محمد) ضمن حالات كينونة الموظف المالكة للكينونة الضعيفة. ومعنى ذلك أن كلَّ حالة من حالات الكينونة الضعيفة تعتمد في وجودها على وجود حالة مقابلة لها في الكينونة القوية.



شكل رقم (٢-٧): مثال لكينونة ضعيفة وارتباطها بالكينونة المالكة.

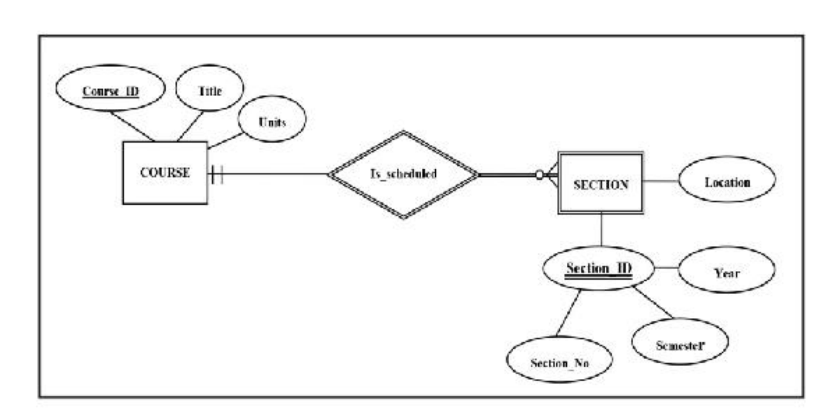
وتُمثَّل الكينونة الضعيفة في مخطط «كينونة - علاقة» بمستطيل ذي حوافٍ مزدوجة الخطوط، كما تُكتَب الخاصية المميزة الجزئية وتحتها خطان كما هو موضحٌ في خاصية اسم الشخص الذي يعوله الموظف. وتُميَّز العلاقة بين الكينونة الضعيفة والكينونة المالكة بالشكل المعيَّن مزدوج الخطوط؛ للدلالة على أن هذه العلاقة هي العلاقة المعرفة التي بدونها لا تُوجَد الكينونة الضعيفة. ويجب أن تكون درجة العلاقة «واحد - متعدد» بين الكينونة المالكة والكينونة الضعيفة، على التوالي. ويعني هذا أن كلَّ حالة من حالات الكينونة الضعيفة يجب أن ترتبط بحالةٍ واحدة فقط في الكينونة المالكة (هي التي أدَّت لوجودها أساساً)، في حين قد ترتبط حالة من حالات الكينونة المالكة بأكثر من حالة في الكينونة الضعيفة. فعلى سبيل المثال: يجب أن يرتبط كلُّ شخص في كينونة الأشخاص الذين يعولهم الموظفون بواحدٍ فقط من الموظفين (هو الموظف الذي يعوله)، في حين قد يعول الموظف أكثر من شخص. أمَّا التعددية الدنيا؛ فتعتمد هنا على قاعدة العمل. ففي المثال الساري قد لا يعول أحد الموظفين أيَّ شخص. وعليه؛ فالتعددية هنا «اختياري - متعدد» كما هو مُوضَّح في مخطط «كينونة - علاقة». أمَّا بالنسبة للمميز الجزئي للكينونة الضعيفة الذي سبق أن أشرنا إليه على أنه للتمييز بين بعض حالات الكينونة الضعيفة؛ فهو في الواقع للتمييز بين الحالات التابعة لكلِّ حالة من حالات الكينونة المالكة. فعلى سبيل المثال: اسم الشخص الذي يعوله الموظف يُعَدُّ مميزاً جزئياً؛ لأنه يميِّز بين الأشخاص الذين يعولهم الموظف الواحد؛ ولكنه ليس مميزاً كاملاً؛ لأنه لا يميِّز بين جميع حالات الكينونة الضعيفة؛ إذ إن بعض الأسماء فيها قد تتكرر.

وكمثالٍ آخر للكينونة الضعيفة؛ لنأخذ قاعدة العمل رقم (5) من قواعد عمل الجامعة الأهلية، ونضع لها النموذج المناسب في مخطط «كينونة - علاقة». تنصُّ قاعدة العمل على التالي:

- **قاعدة العمل (5):** تُنفَّذ (أو تُعقَد) كلُّ مادة دراسية من خلال مجموعةٍ (أو شعبة) دراسية (SeCtion) واحدة أو أكثر في الفصل الدراسي الواحد، أو قد لا تنفذ (أو تعقد) أية مجموعة (أو شعبة) للمادة الدراسية في فصلٍ دراسيٍّ معين، ولكلِّ مجموعة دراسية رمز (SeCtion_ID) يتكون من (رقم المجموعة، والفصل الدراسي المنفذة فيه (Semester)، والسنة الدراسية المنفَّذة فيها (Year)). أمَّا رقم المجموعة (SeCtion_No)؛ فهو رقم (مثل 1، 2، 3، ... إلخ) يميِّز المجموعة عن بقية المجموعات المنفذة للمادة الدراسية نفسها (في نفس الفصل والسنة الدراسيين)؛

ولكنه لا يميِّزها بشكلٍ منفردٍ عن بـقية المجموعات الدراسية المنفذة للمواد الدراسية الأخرى في الجامعة. كما أن لكل مجموعة دراسية موقعاً من مواقع الجامعة لتنفيذها فيه (LoCation).

يُلاحَظ في قاعدة العمل السابقة أنَّ المجموعة الدراسية لا يُوجَد لها خاصية مميزة تميِّزها عن بقية المجموعات الدراسية بـشكلٍ منفرد. فلو أخذنا، على سبيل المثال، مجموعة دراسية ما وافترضنا أن المميِّز للمجموعة الدراسية هو كلُّ الخصائص البسيطة المكونة للخاصية المركبة رمز المجموعة، وهي رقم المجموعة، والفصل الدراسي المنفَّذة فيه، والسنة الدراسية المنفَّذة فيها؛ فإن هذه الخصائص مجتمعةً قد تتكرر لمجموعة دراسية خاصة بمادة دراسية أخرى؛ لأن رقم المجموعة يميِّز بين مجموعات المادة الدراسية الواحدة المنفذة في فصل دراسي من سنة دراسية ما؛ ولكنه لا يميز المجموعة عن مجموعة أخرى منفذة لمادة أخرى في الفصل نفسه من العام الدراسي. ويعني هذا أننا لا نستطيع تمييز المجموعة الدراسية دون معرفة المادة الدراسية التي تتـبعها المجموعة. ومن ثم؛ فإن وجود أية مجموعة دراسية يعتمد على وجود المادة الدراسية التي تتـبعها المجموعة. وبناءً على ذلك؛ فإننا نمثل المجموعة الدراسية (SeCtion) بوصفها كينونةً ضعيفةً، وتكون العلاقة مع المادة الدراسية هي العلاقة المعرَّفة كما هو مُوضَّح في الشكل رقم (2-8).

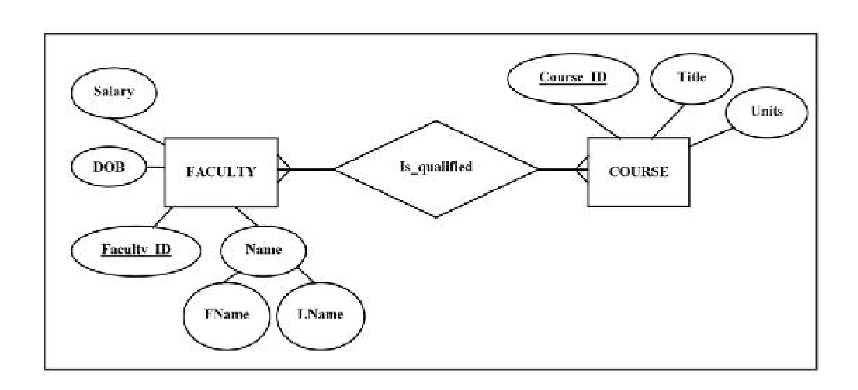


شكل رقم (2–8): تمثيل المجموعة الدراسية ككينونة ضعيفة، وارتباطها بكينونة المادة الدراسية.

2–2–1–2 العلاقات (Relationships):

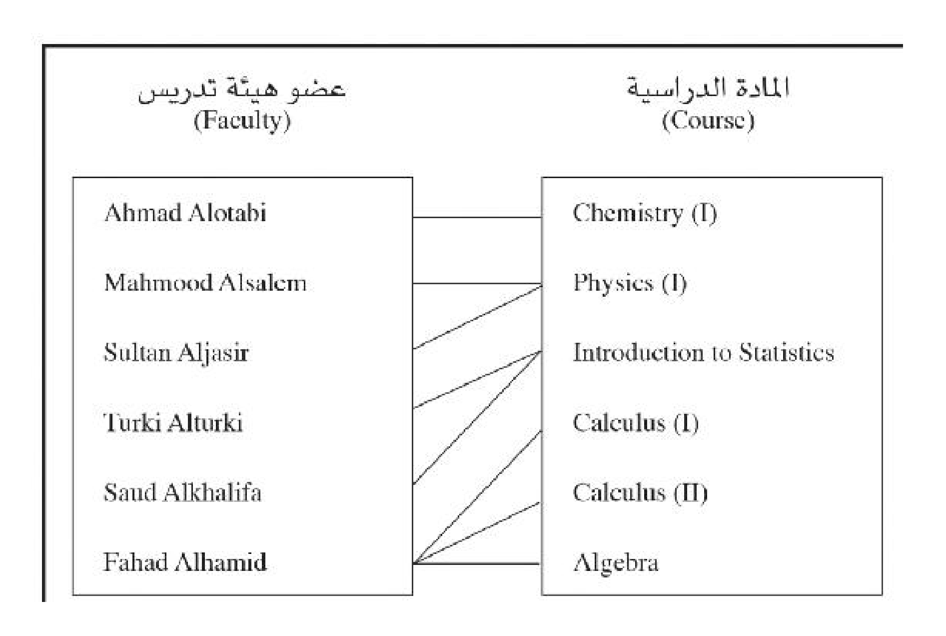
العلاقة هي ارتباطٌ بين حالات فئة كينونة ما بحالات فئة كينونةٍ أخرى وذات أهمية للمنظمة؛ بحيث تسعى إلى تمثيلها ضِمن مخطط «كينونة ـ علاقة» الذي ستُشتقُّ منه وتُبنَى قاعدة البـيانات.

ويعني هذا أن العلاقات في مخطط «كينونة - علاقة» هي الوسيلة التي تمكِّننا من الربط ما بين المكونات المختلفة للمخطط. وعادةً يتمُّ التفريق بين فئة العلاقة (Relationship Type) وحالات العلاقة (Relationship InstanCes)، كما هو الحال لفئة الكينونة وحالات الكينونة. ولإيضاح ذلك لنفترض فئة الكينونة عضو هيئة التدريس (FACULTY)، وفئة الكينونة مادة دراسية (COURSE) مثل تلك الموجودة في الجامعة الأهلية. ولنفترض أننا نرغب في معرفة تأهيل كلِّ عضو هيئة تدريس في الجامعة للمواد التي بإمكانه تدريسها، بمعنى أننا نرغب في معرفة كلِّ المواد الدراسية المؤهل لتدريسها كلُّ عضو هيئة تدريس في الجامعة. في هذه الحالة يتمُّ تعريف فئة علاقة بمُسمَّى «مؤهل لـ» (Is_qualified) بين فئة كينونة أعضاء هيئة التدريس وفئة كينونة المواد الدراسية. ولتمثيل علاقة ما يُستخدَم الشكل المعين مكتوباً بداخله اسمُ العلاقة، وبحيث يكون اسم العلاقة شبه جملة فعلية (Verb Phrase) كما هو مُوضَّح في الشكل رقم (2-9). وتُعَدُّ هذه العلاقة علاقة «متعدد - متعدد»؛ بمعنى أنه قد تكون لكلِّ عضو هيئة تدريس القدرة على تدريس أكثر من مادة دراسية، كما أنه قد يكون للمادة الدراسية الواحدة أكثر من عضو هيئة تدريس مؤهل لتدريسها.



شكل رقم (2–9): تمثيل العلاقات في مخطط «كينونة – علاقة».

ولإيضاح العلاقة السابقة؛ فإن الشكل رقم (2-10) يبيِّن أنَّ كلاً من محمود السالم وسلطان الجاسر مؤهلان لتدريس مادة الفيزياء (1) (I) (PhysiCs))، وأن المواد الثلاث حساب (1) وحساب (2) والجبر (CalCulus (I), CalCulus (II), Algebra)) مؤهل لتدريسها عضو هيئة تدريس واحد هو فهد الحامد.



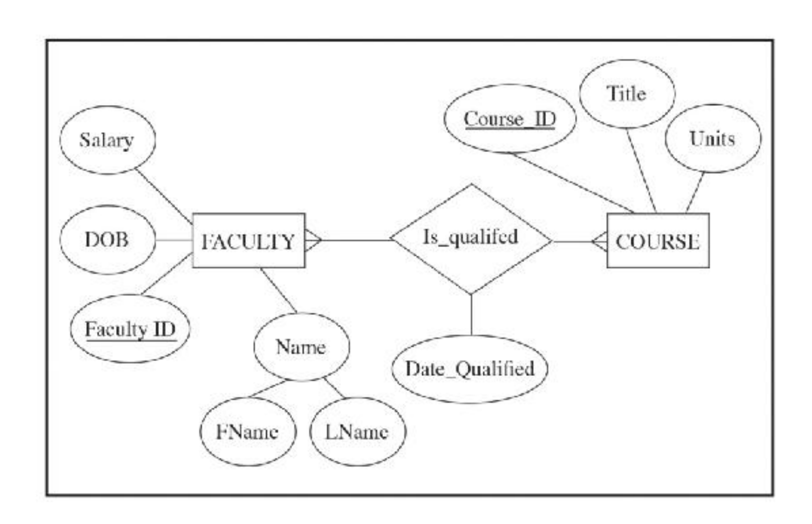
شكل رقم (2–10): مثالٌ لعلاقة أعضاء هيئة التدريس بالمواد الدراسية المؤهلين لتدريسها.

مما سبق يتضح أنَّ فئة العلاقة ما هي إلا ارتباطٌ ذو معنًى بين فئتين من الكينونات (أو أكثر)، وأن فئات العلاقات تمكِّننا من ربط مكونات مخطط «كينونة - علاقة» مع بعضها؛ بحيث نستطيع الإجابة عن تساؤلات لا يمكن الإجابة عنها باستخدام فئات الكينونات فقط. وتُمثَّل فئة العلاقة بالشكل المعين يُكتَب بداخله اسم فئة العلاقة الذي يكون شبه جملة فعلية. كما أنَّ أية حالة من حالات العلاقة تمثل ارتباطاً بين حالات الكينونات التي تربطها فئة العلاقة؛ بحيث يُوجَد حالة واحدة فقط من كلِّ فئة كينونة ترتبط بفئة العلاقة. فكل خط في الشكل رقم (2-10) يمثل حالةً من حالات فئة العلاقة «مؤهل لـ»، وهذه الحالة تمثل ارتباطاً بين حالة من فئة الكينونة «عضو هيئة التدريس» وحالة من حالات فئة الكينونة «مادة دراسية».

2–1–2–2–1 خصائص العلاقة (**Relationship Attributes**):

قد يكون لفئة العلاقة خاصيةٌ أو أكثر كما هو الحال بالنسبة لفئات الكينونات. ففي حال رغبتنا، على سبيل المثال، في تدوين التاريخ الذي تمَّ فيه تأهل عضو هيئة التدريس لتدريس مادة معينة؛ فإنه لا بد أن تكون خاصية تاريخ التأهل مرتبطةً بفئة العلاقة، وليس بأيِّ فئة من الكينونتين اللتين تربط بينهما فئة العلاقة. والسبب وراء ذلك؛ هو أن تاريخ التأهل هو خاصية للعلاقة نفسها

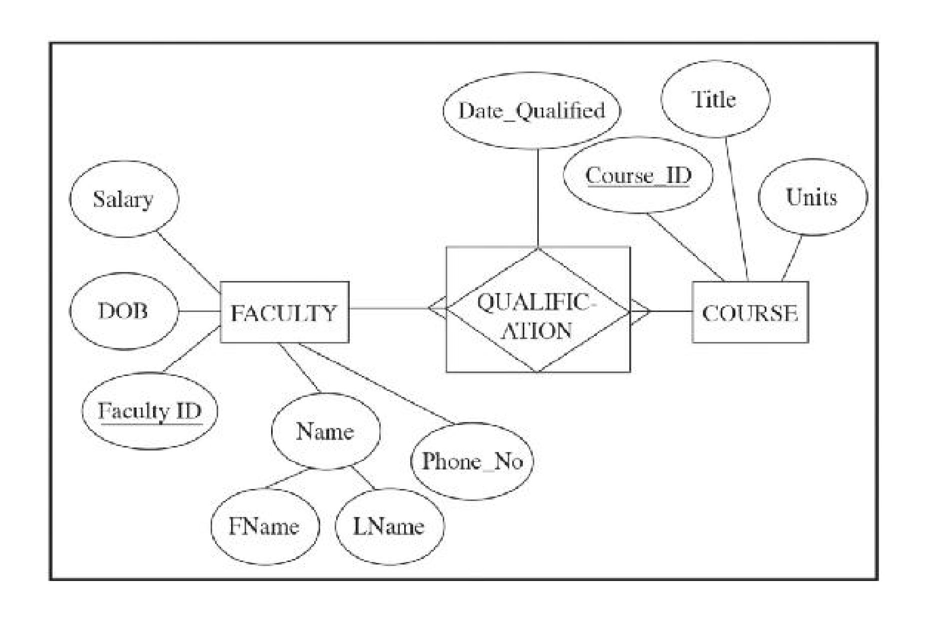
وليست خاصية لأيٍّ من الكينونتين التي تقوم بربطهما ببعض. وهذا التصوُّر يأتي مطابقاً لما نجده من خصائص للعلاقات في حياتنا اليومية؛ فلو نظرنا في العلاقة الزوجية بين رجل وامرأة، على سبيل المثال؛ فإن تاريخ الزواج يُعَدُّ خاصيةً للعلاقة الزوجية بين الطرفين وليس خاصيةً لأيٍّ منهما. كذلك هو الحال بالنسبة لتاريخ المِلكية لعقار ما؛ إذ إن تاريخ تملك العقار يُعَدُّ خاصيةً لعلاقة التملُّك وليس خاصيةً لأيٍّ من المالك أو العقار. وفي مثل هذه الحالات تربط الخاصية (أو مجموعة الخصائص) بفئة العلاقة كما هو مُوضَّح في الشكل رقم (2-11) لفئة العلاقة «مؤهل لـ».



شكل رقم (2–11): تمثيل خصائص العلاقة في مخطط «كينونة – علاقة».

2–1–2–2–2 الكينونة المشاركة (AssoCiative Entity):

إن ارتباط خاصية أو أكثر بفئة علاقة، كما في حالة فئة العلاقة «مؤهل لـ» في الشكل (2-11)، قد تعني أنه من الأنسب أن تُمثَّل فئة العلاقة في مخطط «كينونة - علاقة» على أنها من فئة كينونة. والكينونة المشاركة (AssoCiative Entity) هي فئة علاقة تمَّ تحويلها إلى فئة كينونة، وبذلك فهي تربط ما بين حالات فئتين من الكينونات (أو أكثر)، وترتبط بالخصائص المتعلقة بها. ويُمثَّل شكلُ العلاقة المشاركة بمستطيل بداخله مُعيَّن؛ للدلالة على أن الكينونة قد كانت في الأساس علاقة، ولكن تمَّ تحويلها لعلاقة مشاركة. ويُكتَب بداخل الشكل اسم الكينونة المشاركة؛ بحيث يكون اسماً مشتقاً من اسم فئة العلاقة التي تمَّ تحويلها، كما تبين العلاقة المشاركة في الشكل رقم (2-12) التي تمَّ فيها تحويل فئة العلاقة «مؤهل لـ» (Is_qualified)؛ لتصبح علاقة مشاركة بمُسمَّى «التأهيل» (QualifiCation).



شكل رقم (2–12): تمثيل العلاقة المشاركة في مخطط «كينونة – علاقة».

ويُلاحَظ في المثال السابق عدم وجود فئة علاقة التي تمثل بالشكل المعين في مخطط «كينونة - علاقة» بين الكينونة المشاركة والكينونتين الأخريين؛ وذلك لأن فئة الكينونة المشاركة تمثل العلاقة بين الكينونتين الأخريين، وأنها في الأساس كانت من فئة علاقة. كما يُلاحَظ أن كلتا التعدُّديتين (وهن من نوع متعدد) قد تمَّت إزاحتهما؛ بحيث ينتهيان في الكينونة المشاركة (أو كينونة الربط)؛ عوضاً عن انتهائهما بالكينونتين الأخريين. ومن الأمور التي قد تستدعي تحويل فئة علاقة إلى فئة علاقة مشاركة (أو كينونة ربط) هو توفُّر بعض الشروط التالية (Hoffer et al, 2018):

1- أن تكون مشاركة كلِّ فئة كينونة مرتبطةً بفئة العلاقة من نوعٍ متعدد.

2- أن يكون لفئة العلاقة بعد تحويلها إلى كينونة مشاركة معنى مستقل ومعروف في بيئة عمل المستفيدين، ومن المُستحسَن أن يكون لفئة الكينونة المشاركة مميزٌ يتكون من خاصية واحدة فقط.

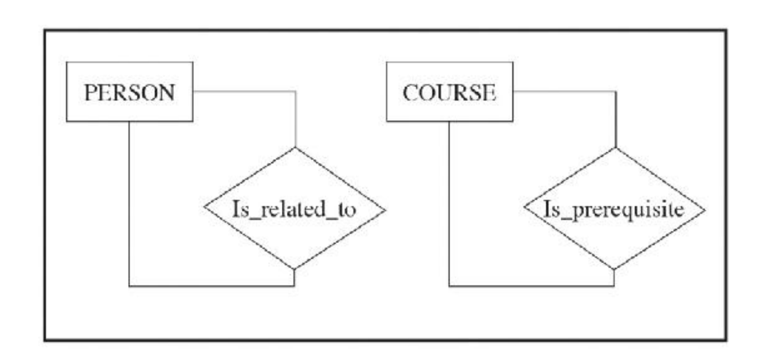
3- أن يكون لفئة العلاقة خاصيةٌ أو أكثر بالإضافة إلى الخاصية التي ستمثل مميزاً للكينونة المشاركة بعد عملية التحويل.

4- أن ترتبط فئة الكينونة المشاركة (بعد عملية تحويل فئة العلاقة إلى كينونة مشاركة) بعلاقات مع كينونات أخرى غير تلك التي أدَّت إلى تكوين فئة الكينونة المشاركة.

2-1-2-2-3 درجة العلاقة (Degree of a Relationship):

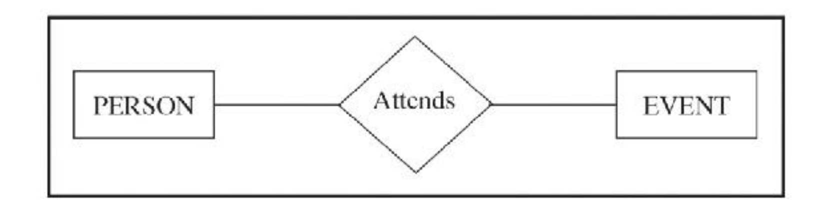
درجة العلاقة هي عددُ فئات الكينونات المرتبطة في فئة العلاقة، ومنها العلاقة الأحادية، والعلاقة الثنائية والعلاقة الثلاثية. وعلى الرغم من أنه يمكن تمثيل علاقات ذات درجات أعلى؛ فإنه قلما تُوجَد مثل هذه العلاقات على أرض الواقع. وفيما يلي أمثلة لكلِّ نوعٍ من درجات العلاقات.

1- علاقة أحادية (Unary Relationship): العلاقة الأحادية؛ هي علاقة تربط بين حالات الكينونة نفسها. وتمثل العلاقة الأولى الموضَّحة في الشكل رقم (2-13) «صلة القرابة» (Is_related_to) بين الأشخاص. ولكون العلاقة تربط بين حالات نفس فئة الكينونة (PERSON)؛ فإن العلاقة أحادية (تشترك فيها فئة كينونة واحدة فقط). فعلى سبيل المثال: قد يكون شخصٌ ما وليكن (ش1) مرتبطاً بعلاقة قرابة مع مجموعة من الأشخاص مثل: (ش2، ش3، ش4)، وكل شخص من (ش2، ش3، ش4) قد يكون مرتبطاً بعلاقات قرابة مع (ش1) وأشخاص آخرين كذلك. كذلك هو الحال بالنسبة للعلاقة الأحادية الثانية الموضَّحة في نفس الشكل التي تمثل المتطلبات الدراسية للمواد الدراسية المختلفة في الجامعة الأهلية؛ إذ إن المادة الدراسية الواحدة قد تكون متطلباً لمواد دراسية أخرى، والمادة الدراسية نفسها قد تتطلب انتهاء الطالب من مجموعة مواد قبل أن يتمكن من التسجيل في المادة. ومن ثم؛ فإن هذه العلاقة تربط بين حالات نفس فئة كينونة المواد الدراسية (COURSE).



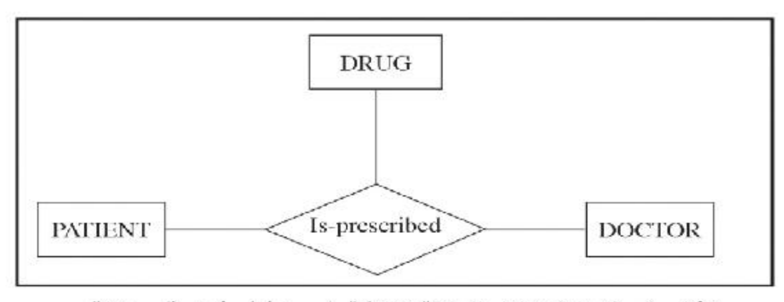
شكل رقم (2-13): تمثيل العلاقة الأحادية في مخطط «كينونة - علاقة».

2- علاقة ثنائية (Binary Relationship): العلاقة الثنائية تربط بين حالات فئتين من الكينونات. وتُمثل العلاقة الثنائية في الشكل رقم (2-14) علاقة حضور الشخص لمناسبة معينة. ولكون علاقة الحضور (Attends) تربط بين الحالات التابعة لنوعين مختلفين من فئات الكينونات، وهما كينونة الشخص (PERSON) وكينونة المناسبة (EVENT)؛ فإن درجة العلاقة ثنائية.



شكل رقم (2–14): تمثيل العلاقة الثنائية في مخطط «كينونة – علاقة».

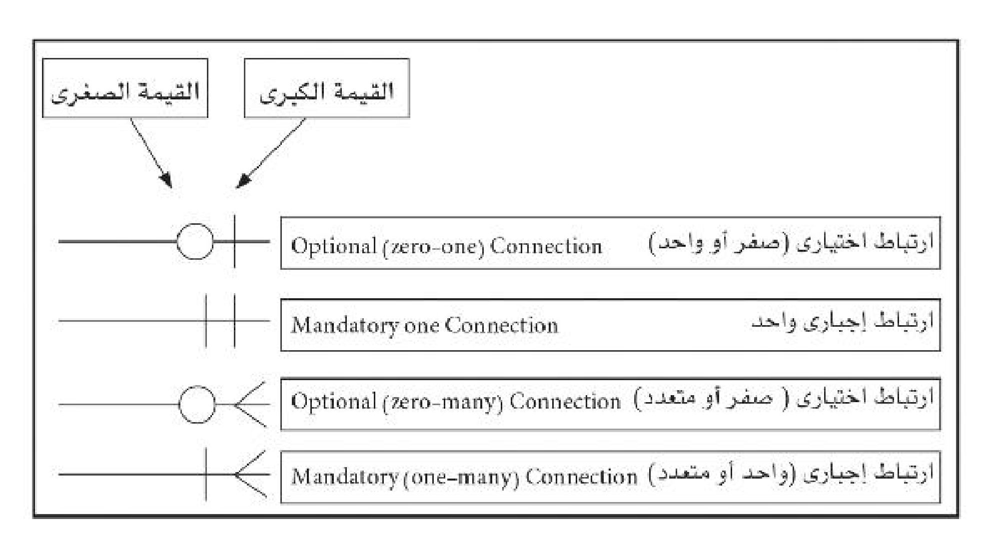
3- علاقة ثلاثية (Ternary Relationship): العلاقة الثلاثية تربط بين حالات ثلاث فئات من الكينونات في وقتٍ واحد. وتُمثِّل العلاقة في الشكل رقم (2-15) علاقة الوصفة الطبية للمريض؛ حيث يشارك في العلاقة ثلاث فئات من الكينونات، وهي: المريض (PATIENT)، والطبيب (DOCTOR)، والدواء (DRUG). ومُعنى «بنفس الوقت» أنه لا يمكن أن تُوجَد حالة من حالات فئة علاقة الوصفة الطبية (Is_presCribed) دون مشاركة حالة من حالات الطبيب، (وهو الذي وصف العلاج) مع حالة من حالات المريض (الذي سيتعاطاه) مع حالةٍ من حالات الدواء (الذي تمَّ وصفه).



شكل رقم (٢-١٥): تمثيل العلاقة الثلاثية في مخطط «كينونة - علاقة».

2–1–2–2–4 قيود التعددية (Cardinality Constraints):

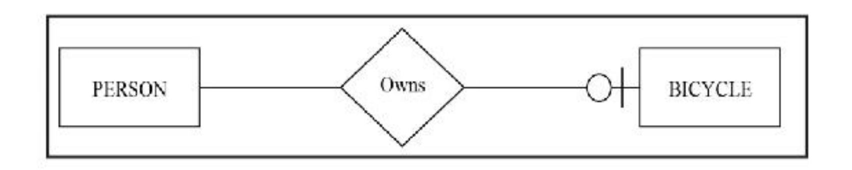
تُحَدِّد قيود التعدُّدية في نموذج «كينونة ـ علاقة» عدد الحالات التي يجب أو يمكن ارتباطها في كينونةٍ ما مع كلِّ حالة في كينونةٍ أخرى. وتُمثّل قيود التعددية من خلال القيمة الصغرى والقيمة الكبرى للعلاقة؛ حيث تمثل القيمة الصغرى للعلاقة أقلَّ عددٍ من الحالات التي يجب أو يمكن ارتباطها في كينونةٍ ما مع كلِّ حالة في كينونة أخرى، في حين تمثل القيمة الكبرى أكبر تعدُّدية للعلاقة أو بمعنى آخر أكبر عددٍ من الحالات التي يجب أو يمكن ارتباطها في كينونة ما مع كلِّ حالة في الكينونة الأخرى. وعند تمثيل قيود التعددية تُستخدَم الرموز الموضَّحة في الشكل رقم (2-16).



شكل رقم (٢-١٦): الرموز المُستخدَمة لتمثيل قيود التعددية في مخطط «كينونة - علاقة».

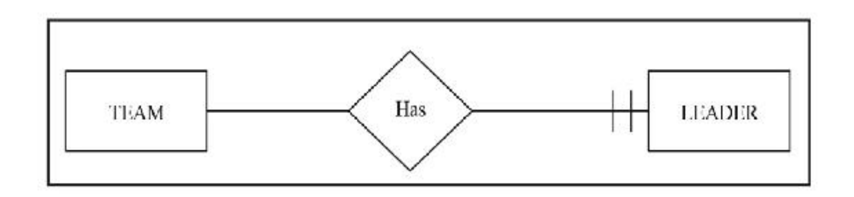
وفيما يلي مثالٌ لكلٍّ من الأنواع الأربعة من قيود التعدُّدية في نموذج «كينونة ـ علاقة»:

1- اختياري واحد (Optional One): تعني العلاقة في الشكل رقم (2-17) أنَّ الحالة الواحدة في كينونة «شخص» (PERSON) قد لا ترتبط بأيَّة حالة في كينونة «دراجة» (BICYCLE)، أو أنها قد ترتبط بحالةٍ واحدةٍ على الأكثر. وبذلك؛ فإن معنى العلاقة يصبح «قد لا يمتلك الشخص الواحد أية دراجة؛ ولكنه في حالة امتلاكه لدراجة؛ فإنه يمتلك دراجةً واحدةً على الأكثر». ولكونه ليس من الضروري أن يمتلك كل شخص لدراجة؛ فإن هذه العلاقة تُعَدُّ اختياريةً.



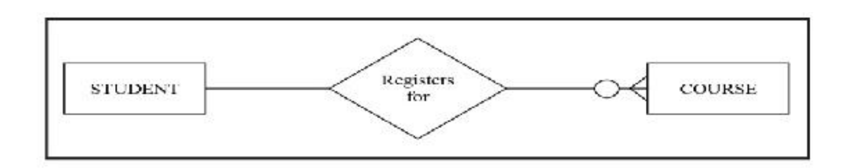
شكل رقم (2–17): تمثيل تعددية اختياري واحد في مخطط «كينونة – علاقة».

2- إجباري واحد (Mandatory One): تعني العلاقة في الشكل رقم (2-18) أنَّ الحالة الواحدة في كينونة «فريق» (TEAM) يجب أن ترتبط بحالةٍ واحدة في كينونة «قائد» (LEADER)، وحالة واحدة فقط (كحدٍّ أعلى). وبذلك؛ فإن معنى العلاقة بين كينونة الفريق وكينونة القائد يُصبح «لكل فريق قائد، وقائد الفريق واحد فقط». ولكونه
لا بد أن يكون لكل فريق قائد واحد (حدًّا أدنى)؛ فإن العلاقة بين كينونة فريق وكينونة قائد علاقة إجبارية.



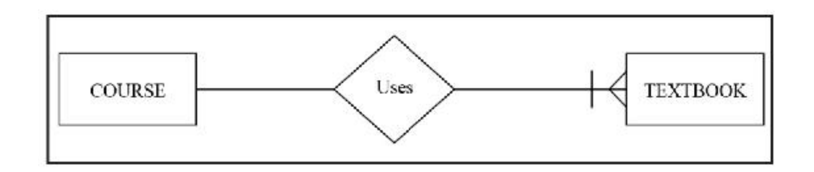
شكل رقم (2–18): تمثيل تعددية إجباري واحد في مخطط «كينونة – علاقة».

3- اختياري متعدِّد (Optional Many): تعني العلاقة في الشكل رقم (2-19) أن الحالة الواحدة في كينونة «طالب» (STUDENT)، قد لا ترتبط بأية حالة في كينونة «مادة دراسية» (COURSE)، أو أنها قد ترتبط بـأكثر من حالة واحدة. وبذلك؛ فإن معنى العلاقة يصبح «قد لا يُسجّل الطالب الواحد في أية مادة دراسية، أو أنه قد يُسجّل في أكثر من مادة دراسية». ولكونه ليس من الضروري أن يسجل الطالب في أية مادة؛ فإن العلاقة اختيارية. كما أن العلاقة متعددةٌ؛ لكون الطالب قد يسجل في أكثر من مادة دراسية. وبذلك تصبح العلاقة اختياريةً متعددةً.



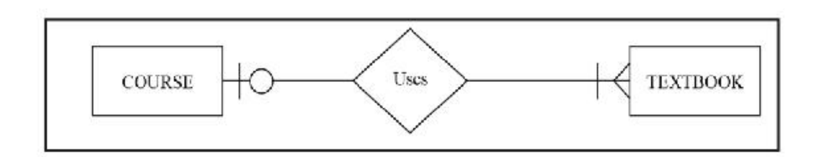
شكل رقم (2–19): تمثيل تعددية اختياري متعدد في مخطط «كينونة – علاقة».

4- إجباري مُتعدِّد (Mandatory Many): تعني العلاقة في الشكل رقم (2-20) أن الحالة الواحدة في كينونة «مادة دراسية» (COURSE)؛ يجب أن ترتبط بحالة واحدة في كينونة «كتاب دراسي» (TEXTBOOK) على الأقل، وقد ترتبط بأكثر من حالة في كينونة كتاب دراسي. وبذلك؛ فإن معنى العلاقة ما بين كينونة المادة الدراسية والكتاب الدراسي يصبح «لكلِّ مادة دراسية كتاب واحد على الأقل، وقد يكون للمادة الدراسية أكثر من كتاب دراسي». ولكونه لا بد أن يكون لكل مادة دراسية كتاب واحد (حدًّا أدنى)؛ فإن العلاقة بين كينونة مادة دراسية وكينونة كتاب دراسي علاقة إجبارية. كما أن العلاقة متعددةٌ؛ لكونه قد يكون للمادة الدراسية الواحدة أكثر من كتاب دراسي واحد.



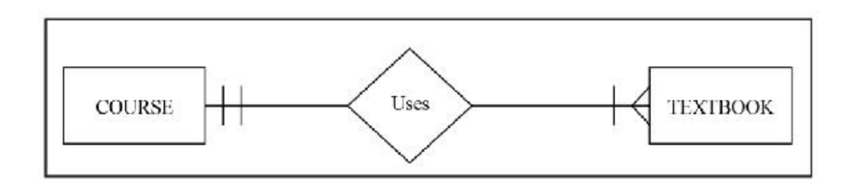
شكل رقم (2–20): تمثيل تعددية إجباري متعدد في مخطط «كينونة – علاقة».

ولأن العلاقة بين الكينونات المرتبطة بها علاقة متبادلة؛ فإنه يجب إدراج قيود تعددية العلاقة في كل اتجاهاتها، بمعنى أننا قد نظرنا في الأمثلة السابقة إلى العلاقات في اتجاه واحد فقط وهو من الجهة اليسرى من العلاقة إلى الجهة اليمنى. ولإدراج تعددية العلاقة في الاتجاه الآخر للمثال الأخير، على سبيل المثال، إذا ما علمنا أن قاعدة العمل تنصُّ على أنه «قد لا يُخصَّص الكتاب الدراسي لمادة دراسية، ولكنه قد يُستخدَم من قِبل مادة دراسية واحدة (على الأكثر)»؛ فإن العلاقة تصبح اختيارية واحدة كما هو مُوضَّح في الشكل رقم (2-21).



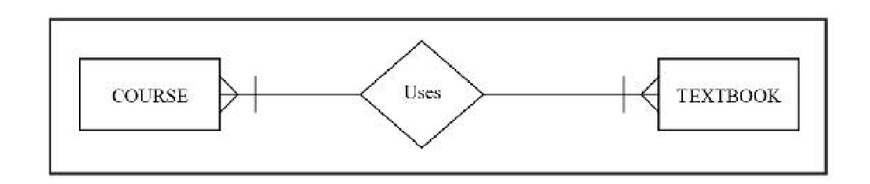
شكل رقم (2–21): تمثيل تعددية العلاقة اختياري واحد وإجباري متعدد.

أما إن نصَّت قاعدة العمل على أن «كلّ كتاب دراسي مخصصٌ لمادة دراسية واحدة فقط»؛ فإن العلاقة تصبح إجبارية واحدة كما في الشكل (2–22).



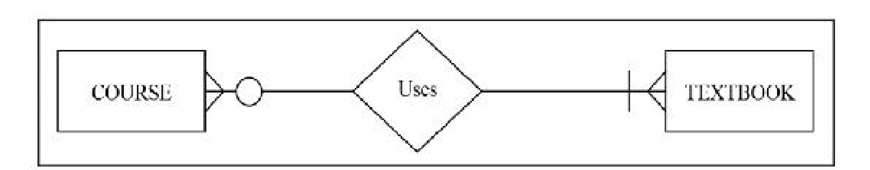
شكل رقم (2–22): تمثيل تعددية العلاقة إجباري واحد وإجباري متعدد.

وفي حال نصَّت قاعدة العمل على أنَّ «كلّ كتاب دراسي مخصصٌ لمادةٍ دراسية واحدة على الأقل، وقد يُخصَّص الكتاب لأكثر من مادة دراسية»؛ فإن العلاقة تصبح إجباريةً متعددةً كما في الشكل رقم (2-23).



شكل رقم (2–23): تمثيل تعددية العلاقة إجباري متعدد وإجباري متعدد.

والاحتمال الأخير المتبقي للعلاقة؛ هو في حال نصَّت قاعدة العمل على أن «الكتاب الواحد قد لا يُخصَّص لأية مادة دراسية، وقد يُخصَّص لأكثر من مادة دراسية». في هذه الحالة تصبح العلاقة اختياريةً متعددةً كما في الشكل رقم (2-24).



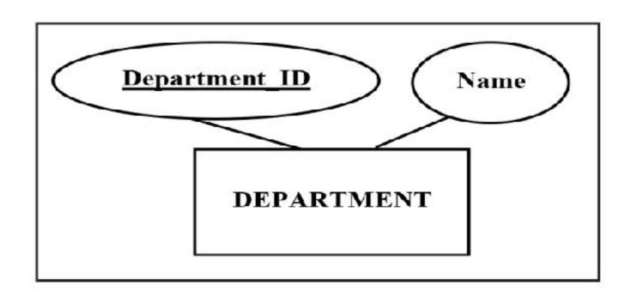
شكل رقم (2–24): تمثيلُ تعددية العلاقة اختياري متعدد وإجباري متعدد.

2–1–2–3 حالة تطبيقية:

بعد استعراضنا للمفاهيم الأساسية المكوِّنة لنموذج «كينونة - علاقة»، نعود لقواعد العمل المعمول بها في الجامعة؛ بهدف رسم مخطط «كينونة - علاقة» الذي يمثلها.

1- يُوجَد في الجامعة عددٌ من الأقسام الدراسية، ولكلِّ قسمٍ (Department) من الأقسام العلمية رمز (Department_ID) يميِّزه عن بقية الأقسام، واسم (Name).

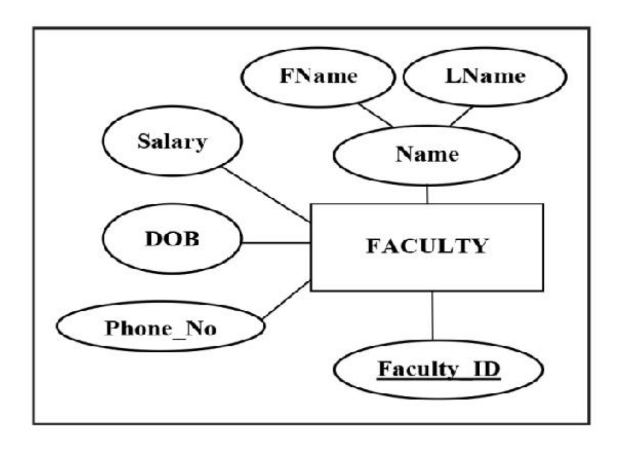
لقد سبق أن مثلنا قاعدة العمل هذه بفئة كينونة القسم الدراسي والخواص التابعة لها مع تحديد الخاصية المميزة، وكان تمثيلنا للقسم الدراسي كما في الشكل رقم (2-25).



شكل رقم (2–25): تمثيل قاعدة العمل الأولى للجامعة الأهلية في مخطط «كينونة – علاقة».

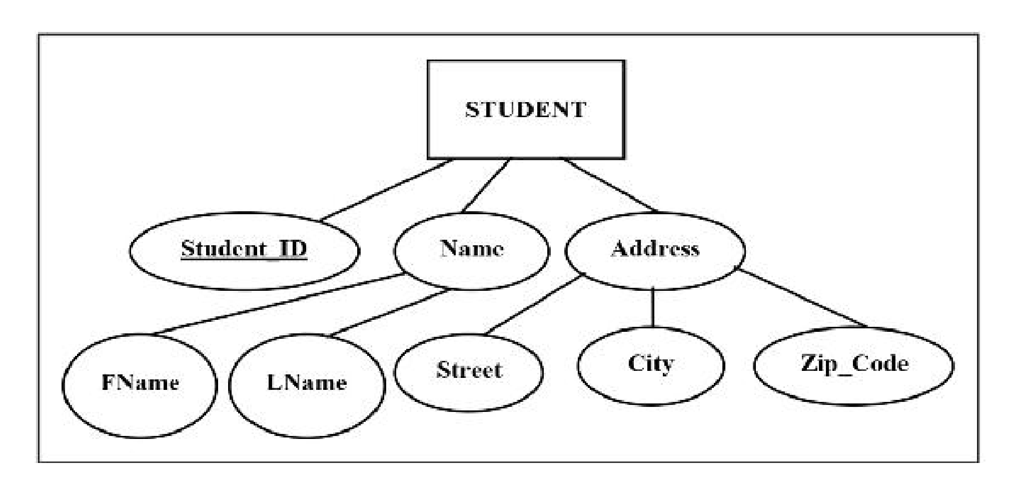
2- يعمل في الجامعة عددٌ من أعضاء هيئة التدريس، ولكلِّ عضو هيئة تدريس (FaCulty) رمز (FaCulty_ID) يميِّزه عن بقية أعضاء هيئة التدريس، واسم (Name) يتكون من (الاسم الأول (FName) واسم العائلة (LName))، وراتب شهري (Salary)، وتاريخ ميلاد (Date of Birth (DOB))، ورقم هاتف (Phone_No).

لقد سبق أنَّ مثلنا قاعدة العمل هذه بفئة كينونة عضو هيئة التدريس والخواص التابعة لها مع تحديد الخاصية المميزة، وكان تمثيلنا لفئة الكينونة كما في الشكل رقم (2-26).



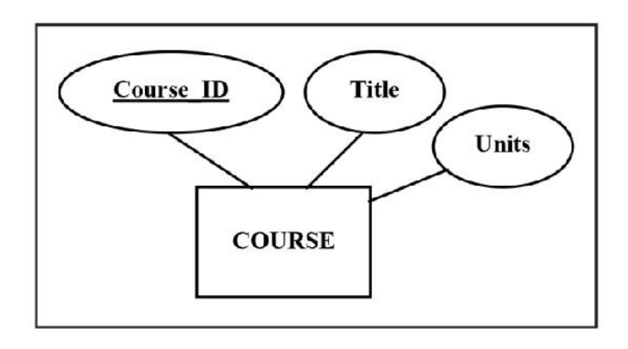
شكل رقم (2–26): تمثيل قاعدة العمل الثانية للجامعة الأهلية في مخطط «كينونة – علاقة».

3- يدرس في الجامعة عددٌ من الطلاب، ولكل طالب (Student) رمز (Student_ID) يميّزه عن بقية الطلاب في الجامعة، واسم (Name) يتكون من (الاسم الأول (FName) واسم العائلة (LName))، وعنوان بريدي (Address) يتكون من (اسم الشارع (Street)، واسم المدينة (City)، والرمز البريدي (Zip_Code)).



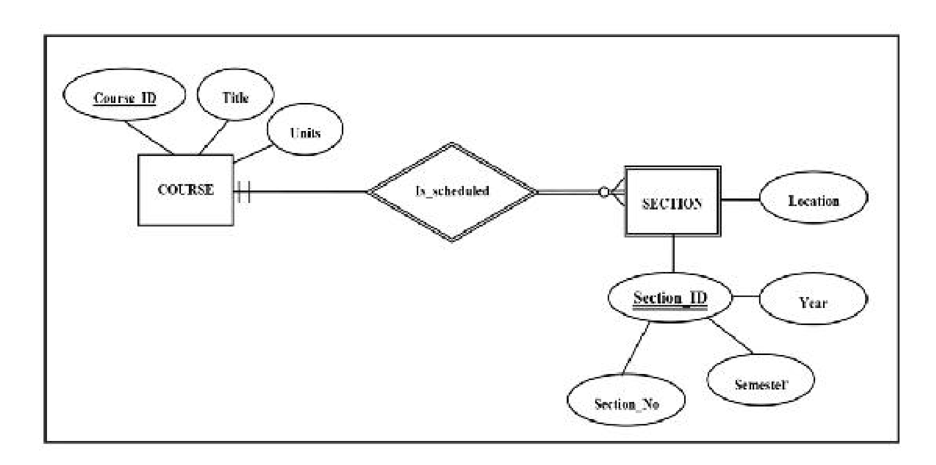
شكل رقم (2–27): تمثيل قاعدة العمل الثالثة للجامعة الأهلية في مخطط «كينونة – علاقة».

4- تُنفِّذ الجامعة مجموعةً من المواد الدراسية، ولكلِّ مادة دراسية (Course) رمز (Course_ID) يميزها عن بقية المواد الدراسية التي تنفذها الجامعة، واسم (Title)، وعدد وحدات (أو ساعات) دراسية (Units).



شكل رقم (2–28): تمثيل قاعدة العمل الرابعة للجامعة الأهلية في مخطط «كينونة – علاقة».

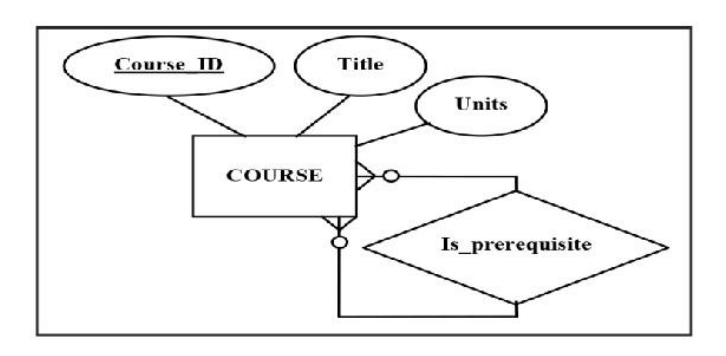
5- تنفذ (أو تعقد) كلُّ مادة دراسية من خلال مجموعة (أو شعبة) دراسية (SeCtion) واحدة أو أكثر في الفصل الدراسي الواحد، أو قد لا تنفذ (أو تعقد) أية مجموعة (أو شعبة) للمادة الدراسية في فصل دراسي معين، ولكل مجموعة دراسية رمز (SeCtion_ID) يتكون من (رقم المجموعة، والفصل الدراسي المنفذة فيه (Semester)، والسنة الدراسية المنفذة فيها (Year)). أمَّا رقم المجموعة (SeCtion_No)؛ فهو رقم (مثل: 1، 2، 3، ... إلخ) يميز المجموعة عن بقية المجموعات المنفذة للمادة الدراسية نفسها (وفي نفس الفصل والسنة الدراسيين)؛ ولكنه لا يميزها بشكلٍ منفرد عن بـقية المجموعات الدراسية المنفذة للمواد الدراسية الأخرى في الجامعة. كما أن لكلِّ مجموعة دراسية موقع من مواقع الجامعة لتنفيذها فيه (LoCation).



شكل رقم (2–29): تمثيل قاعدة العمل الخامسة للجامعة الأهلية في مخطط «كينونة – علاقة».

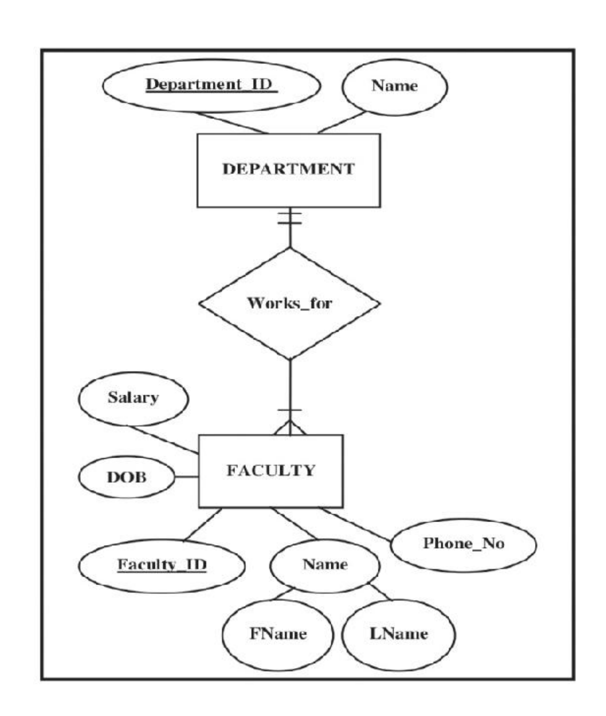
6- قد يكون للمادة الدراسية الواحدة مجموعةٌ من المتطلبات الدراسية، أو قد لا يكون للمادة الدراسية أية متطلبات دراسية. كما أن المادة الدراسية الواحدة قد تكون متطلباً لأكثر من مادة

دراسية، أو قد لا تكون متطلباً لأية مادة دراسية.



شكل رقم (2–30): تمثيل قاعدة العمل السادسة للجامعة الأهلية في مخطط «كينونة – علاقة».

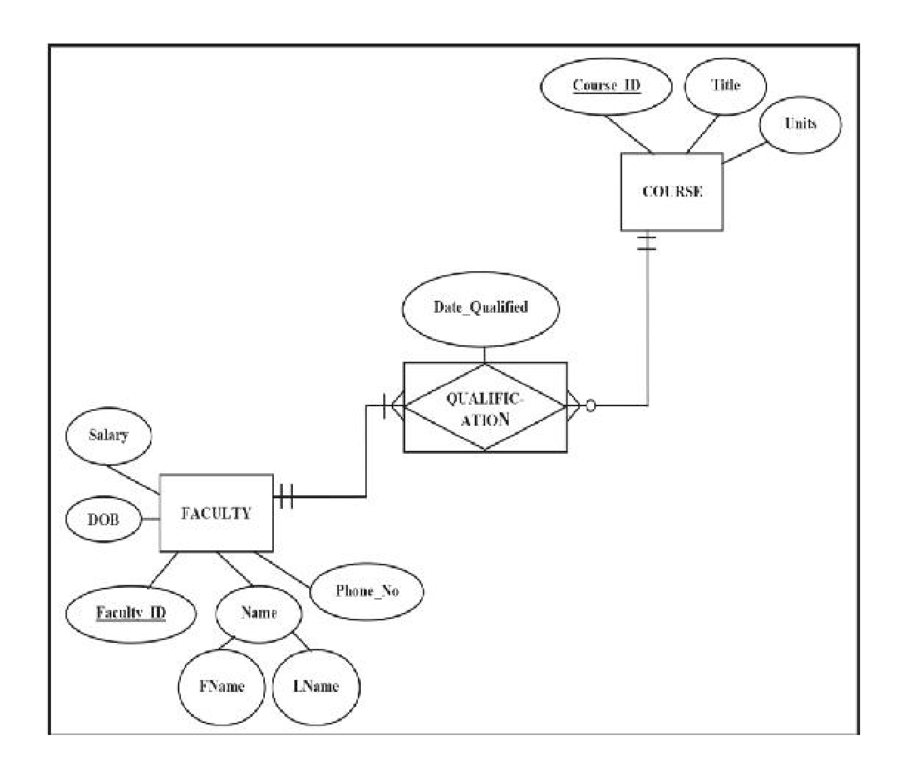
7- يعمل في (works for) كل قسم من أقسام الجامعة عضو هيئة تدريس واحد أو أكثر، وكل عضو من أعضاء هيئة التدريس يعمل في قسم دراسي واحد فقط.



شكل رقم (2–31): تمثيل قاعدة العمل السابعة للجامعة الأهلية في مخطط «كينونة – علاقة».

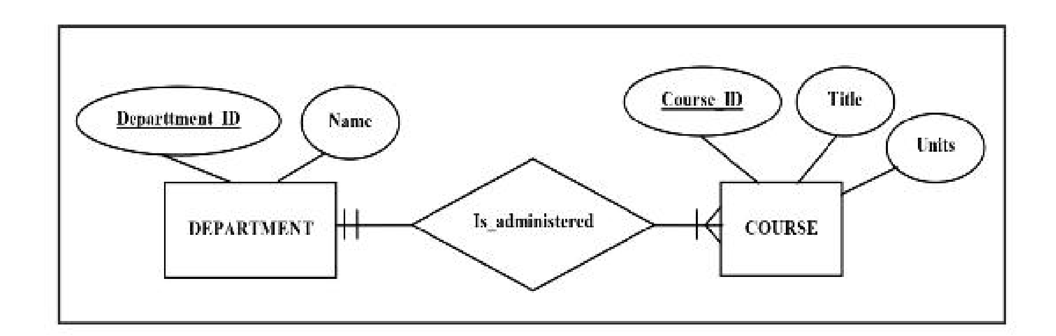
8- كل عضو هيئة تدريس في الجامعة مؤهل (Qualified) لتدريس مادة دراسية واحدة على الأقل، وقد يتوفر للمادة الدراسية الواحدة أكثر من عضو هيئة تدريس مؤهلٍ لتدريسها، وقد لا يُوجَد من أعضاء هيئة التدريس مَنْ هو مؤهل لتدريس المادة.

9- عندما يتأهل عضو هيئة التدريس لتدريس مادةٍ ما لأول مرة، يكون هنالك تاريخ لتأهيله (QualifiCation Date) يُحدِّد تاريخ تأهل عضو هيئة التدريس لتدريس المادة الدراسية.



شكل رقم (2–32): تمثيل قاعدة العمل الثامنة والتاسعة للجامعة الأهلية في مخطط «كينونة – علاقة».

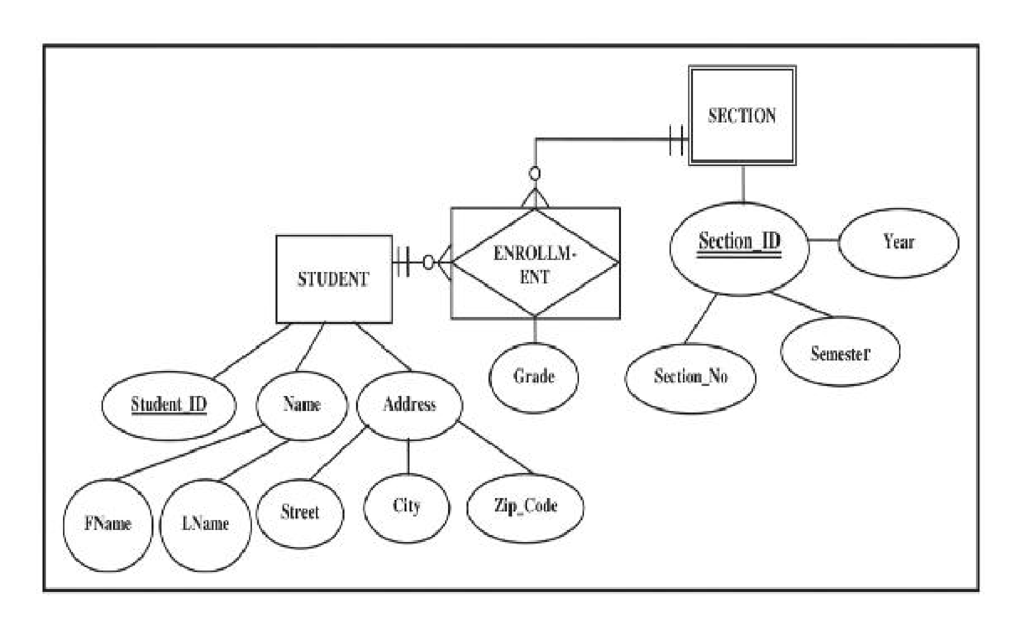
10- تُدار (Administered) كلُّ مادة دراسية من قِبل قسمٍ دراسيٍّ واحد من أقسام الجامعة، ويُدِير كلُّ قسم مادة دراسية واحدة على الأقل.



شكل رقم (2–33): تمثيل قاعدة العمل العاشرة للجامعة الأهلية في مخطط «كينونة – علاقة».

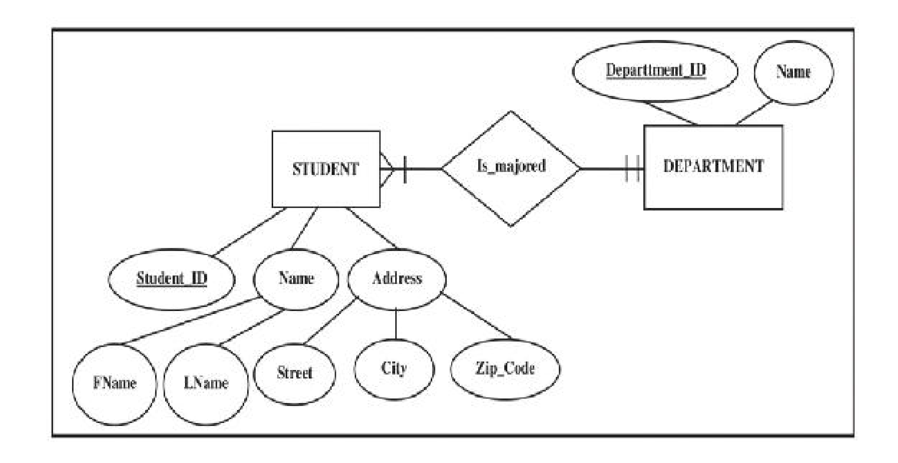
11- قد يسجل (Enrolls) الطالب الواحد في أكثر من مجموعة (أو شعبة) دراسية أو قد لا يسجل في أية مجموعة (أو شعبة) دراسية، والمجموعة (أو الشعبة) الدراسية الواحدة قد لا يسجل فيها أيُّ طالب أو قد يسجل فيها أكثر من طالب.

12- عندما يُسجّل طالب في مجموعة دراسية تكون له درجة (Grade) تعطى عند انتهائه من الدراسة في المجموعة.



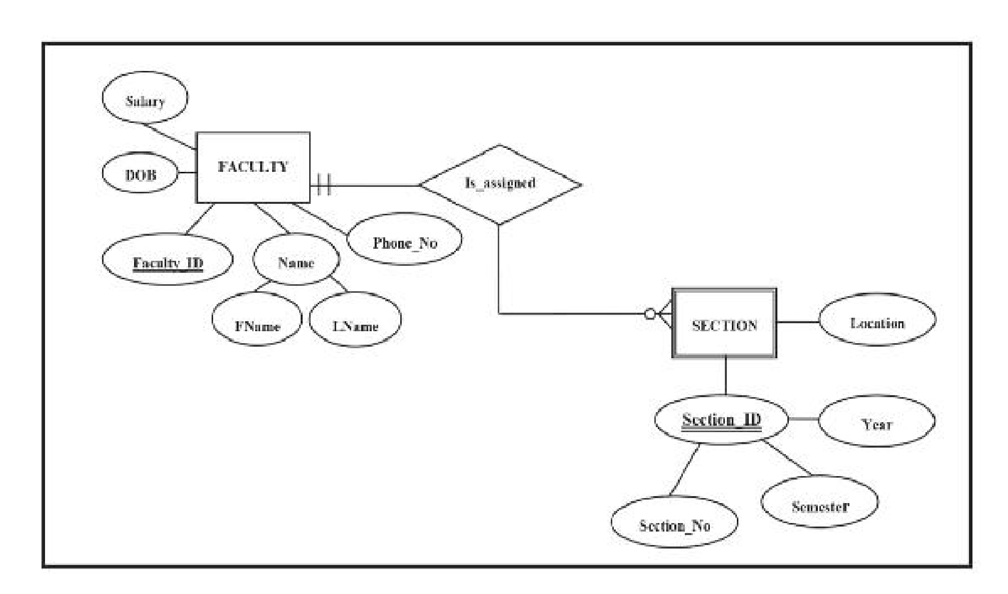
شكل رقم (2–34): تمثيل قاعدة العمل الحادية عشرة والثانية عشرة
للجامعة الأهلية في مخطط «**كينونة – علاقة**».

13- يتخصص كلُّ طالب (Majors) في قسم دراسي واحد فقط، ويتخصص في القسم الدراسي الواحد أكثر من طالب.



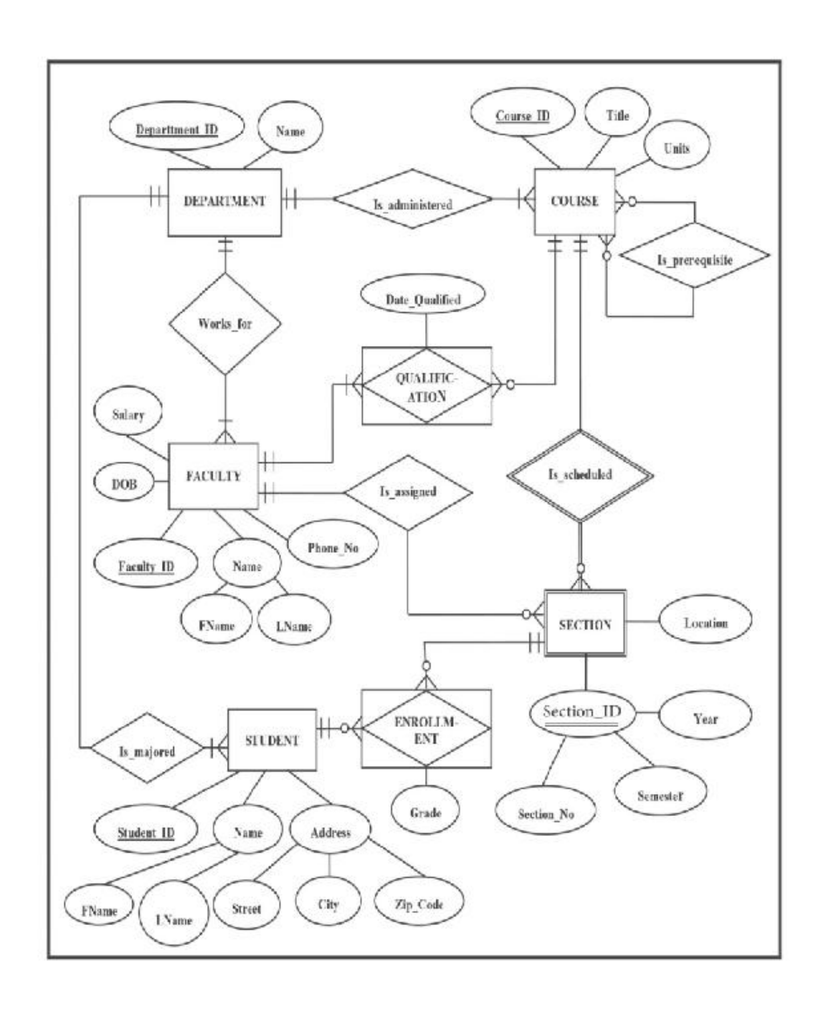
شكل رقم (2–35): تمثيل قاعدة العمل الثالثة عشرة للجامعة الأهلية في مخطط «كينونة – علاقة».

14- يُكلَّف (Assigned) كل عضو هيئة تدريس بتدريس مجموعة (أو شعبة) دراسية واحدة أو أكثر، وقد لا يكلف عضو هيئة التدريس بأية مجموعة (أو شعبة) دراسية، والمجموعة (أو الشعبة) الدراسية الواحدة تكلف لعضو هيئة تدريس واحد فقط.



شكل رقم (2–36): تمثيل قاعدة العمل الرابعة عشرة للجامعة الأهلية في مخطط «كينونة – علاقة».

ويمثل الشكل رقم (2-37) كامل مخطط «كينونة - علاقة» وفق قواعد العمل المعمول بها في الجامعة الأهلية المذكورة أعلاه.



شكل رقم (2–37): كامل مخطط «كينونة – علاقة» للجامعة الأهلية وفق قواعد العمل المفترضة.

الفصل الثالث

نموذج كينونة – علاقة المطور

إن نموذج كينونة - علاقة الذي تم استعراض مكوناته في الفصل السابق قد لاقى قبولاً كبيراً من مُصمِّمي نظم قواعد البيانات منذ بداية ظهوره في منتصف السبعينيات الميلادية؛ وذلك لقدرته على نمذجة غالبية قواعد العمل المعمول بها في المنظمات الحديثة؛ غير أن النموذج بشكله المبدئي لا يستطيع أن يقوم بنمذجة كافة قواعد العمل المعمول بها في غالبية المنظمات حالياً. وقد حدا هذا بالباحثين إلى تطوير المكونات الرئيسية للنموذج؛ حتى يتمكن من نمذجة أكبر قدرٍ ممكنٍ من قواعد العمل المعمول بها في المنظمات المختلفة. ويستعرض هذا الفصل من الكتاب نموذج «كينونة - علاقة المطور» (Entity-Relationship Model EnhanCed). ومن أهمِّ المكونات الرئيسية في النموذج المطور هي «علاقة الأنواع الرئيسية والأنواع الفرعية» (Supertype/Subtype Relationship) ومبدأ «التجميع» (Aggregation).

3–1 الأنواع الرئيسية والأنواع الفرعية (**Supertype/Subtype**) في نموذج كينونة– علاقة المطور:

إن من أهمِّ مكونات نموذج كينونة - علاقة المطور؛ هو «الأنواع الرئيسية والأنواع الفرعية». ويمكِّننا هذا المكون الجديد من نمذجة نوعٍ عام واحد من فئة كينونة، يُسمَّى «النوع الرئيسي»؛ بحيث يتمُّ تقسيمه إلى عدة أنواع مخصَّصة، تُسمَّى «الأنواع الفرعية». ويتكون النوع الرئيسي، كما هو الحال في فئات الكينونات الأخرى، من مجموعة من الخصائص المشتركة في كافة الكينونات الفرعية المخصصة من النوع الرئيسي. كما أن النوع الرئيسي قد يرتبط بعلاقات مع فئات كينونات أخرى. ويتمُّ توريث كلٍّ من خصائص النوع الرئيسي والعلاقات التي يرتبط بها

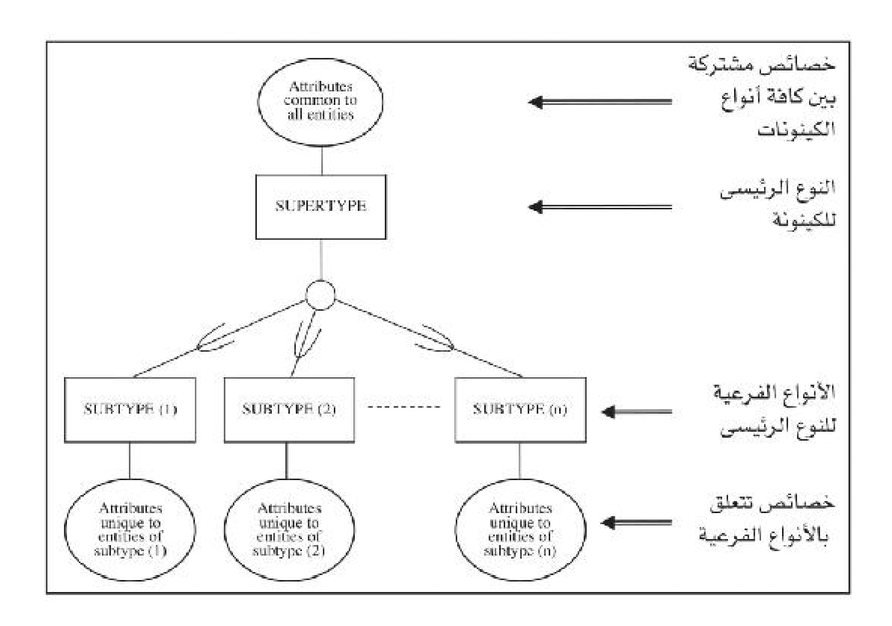
لكلٍّ نوع من الأنواع الفرعية، هذا بالإضافة إلى ما قد يكون للأنواع الفرعية من خصائص (أو علاقات) تميِّزها عن الأنواع الفرعية الأخرى التي تتبع لنفس النوع الرئيسي.

إن الأنواع الفرعية ما هي إلا مجموعات جزئية لأحد الأنواع الرئيسية. فعلى سبيل المثال، يمكن تمثيل فئة كينونة «طالب» (STUDENT)؛ بحيث يتفرع منها نوعان فرعيان، هما: فئة طلبة «الدراسات العليا» (Graduate_Students) وفئة «طلبة البكالوريوس» (Undergraduate_Students). ويُلاحَظ في كلا النوعين الفرعيين أن لهما خصائص مشتركة كما أن كلاً منهما له خصائص لا يشترك فيها مع النوع الفرعي الآخر. فعلى سبيل المثال: رقم الطالب، وعنوان الطالب، ورقم هاتف الطالب، تمثل بعض الخصائص المشتركة لكافة الطلبة بغض النظر عن كونهم من طلبة الدراسات العليا أو طلبة البكالوريوس. على النقيض من ذلك؛ فإن فئة طلبة الدراسات العليا قد يكون لها خصائص تميِّزها عن طلبة دراسات البكالوريوس، مثل: رقم مكتب الطالب (إذ إن غالبية الجامعات توفر مكاتب لطلبة الدراسات العليا)، أو علاقات خاصة بهذه الفئة، مثل ارتباطها بكينونة أعضاء هيئة التدريس لتمثيل علاقة المشرف الدراسي لمجال بحث الطالب. والسؤال الذي قد يُطرَح، لماذا لا نمثل كلَّ فئة من الكينونات الفرعية السابقة بشكلٍ أبسط؛ من خلال نمذجتها على أساس أنها فئات لكينونات مختلفة؟ إن الإجابة عن هذا التساؤل ليست ببساطة السؤال ذاته؛ لأنها تعتمد على ما تريده المنظمة من النموذج؛ إذ إنه من التحديات التي تواجه عملية نمذجة البيانات التعرُّف على الكينونات المتشابهة فيما بينها بشكلٍ كبيرٍ وتمثيلها بشكلٍ واضحٍ ضِمن فئة واحدة. وإن استخدام الأنواع الرئيسية تمكِّننا من ذلك، وفي الوقت نفسه تمكِّننا من تخصيص بعض الأنواع الفرعية التي تتميز بخصائص لا تشترك فيها مع الأنواع الفرعية الأخرى، خاصةً إذا كانت هذه الأنواع الفرعية ذات معنًى في بيئة المنظمة، مثل طلبة الدراسات العليا وطلبة البكالوريوس، ويجب إيضاحها ضِمن نموذج البيانات.

3–1–1 المفاهيم الأساسية والرموز المستخدَمة في الأنواع الرئيسية والأنواع الفرعية:

يوضِّح الشكل رقم (3-1) الرموز الأكثر شيوعاً في تمثيل الأنواع الرئيسية والأنواع الفرعية (Hoffer et al, 2018)؛ إذ يتصل النوع الرئيسي بخط مستقيم مع دائرة يتفرع منها خط مستقيم لكلِّ نوع فرعي من أنواع النوع الرئيسي. أما الرمز (U) الموضوع على الخط الواصل بين نقطة التفرع (وهي الدائرة) إلى كلِّ نوع فرعي؛ فيُقصَد منها أن حالات النوع الفرعي تمثل مجموعةً

جزئيةً من حالات النوع الرئيسي؛ إضافةً إلى كونها تبيّن اتجاه العلاقة بين النوع الرئيسي والنوع الفرعي. وترتبط الخصائص المشتركة لكافة أنواع الكينونات، ويتضمَّن ذلك المميّز (أو المعرف) (Identifier)، بالنوع الرئيسي. أمَّا الخصائص المتعلقة بنوع فرعي ما دون غيره من الأنواع الفرعية؛ فترتبط بالنوع الفرعي نفسه.



شكل رقم (3–1): الرموز الأكثر شيوعاً في تمثيل الأنواع الرئيسية والأنواع الفرعية.

ولتوضيح مفهوم الأنواع الفرعية والأنواع الرئيسية، لنفترض المثال البسيط التالي الذي يُعَدُّ من أكثر الأمثلة شيوعاً، وهو وجود منظمة يعمل فيها ثلاثة أنواع من الموظفين، وهم: موظفون يعملون بأجر الساعة (Hourly Employees)، وموظفون مثبتون على وظائف رسمية بأجر شهري (Salaried Employees)، وموظفون استشاريون يعملون وفق عقود تُحدِّد أجرهم (Consultant Employees). ومن الخصائص المهمة للأنواع الثلاثة من الموظفين، ما يلي.

- موظفو أجر الساعات (Hourly Employees): رقم الموظف (Employee_No)، واسم الموظف (Name)، وعنوان الموظف (Address)، وتاريخ التعيين (Date_Hired)، وأجر الساعة (Hourly_Rate).

- موظفو الأجر الشهري (Salaried Employees): رقم الموظف (Employee_No)، واسم الموظف (Name)، وعنوان الموظف (Address)، وتاريخ التعيين (Date_Hired)،

والمرتب الشهري (Monthly_Salary).

- موظفو الأجر وفق عقود استشارية (Consultant Employees): رقم الموظف (Employee_No)، واسم الموظف (Name)، وعنوان الموظف (Address)، وتاريخ التعيين (Date_Hired)، ورقم العقد (ContraCt_No)، والتكلفة الاستشارية (Billing_Rate).

ويُلاحَظ أن الفئات الثلاث من الموظفين يشتركون في خصائص مشتركة، مثل: رقم الموظف، وعنوان الموظف. كما أنَّ كلَّ فئة من الموظفين ترتبط بخاصية أو أكثر لا ترتبط فيها الفئتان الأخريان من فئات الموظفين، مثل: خاصية أجر الساعة بالنسبة لفئة الموظفين بأجر الساعة. وعندما نحاول نمذجة بيانات هذا المثال وفق نموذج كينونة - علاقة، يمكن النظر في ثلاثة خيارات لنمذجته، وهي كالتالي:

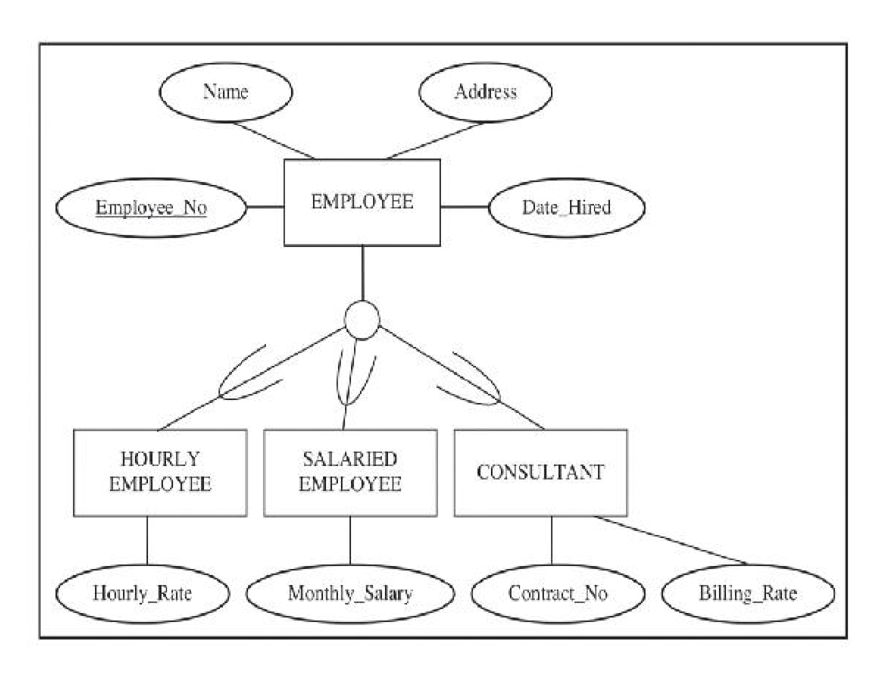
1- تمثيل الفئات الثلاث من الموظفين؛ من خلال فئة كينونة واحدة هي كينونة الموظفين (EMPLOYEE). وعلى الرغم من بساطة هذا النموذج؛ فإن العيب الرئيسي لهذا الأسلوب هو أن كينونة الموظفين يجب أن تحتوي على كافة خصائص الفئات الثلاث من الموظفين. وعند عدم انطباق خاصيةٍ ما على إحدى حالات الموظفين؛ فتترك قيمتها غائبة (Null). وعندما تتم نمذجة هذا المثال وفق هذه الطريقة؛ فإنها ستعقد عملية كتابة التطبيقات على قاعدة البيانات؛ لأن كل تطبيق يجب أن يُكتَب بطريقة يستطيع من خلالها التمييز بين الفئات الثلاث من الموظفين والتعامل معها بشكلٍ سليم، هذا بالإضافة إلى ما قد ينتج عن هذه الطريقة من ضياع للمساحة التخزينية؛ نتيجة للحقول غير المستخدَمة من قِبل كلِّ كينونة من كينونات الفئة.

2- تمثيل كلِّ فئة من فئات الموظفين من خلال فئة كينونة خاصة بها؛ بحيث ينتج عن هذه الطريقة ثلاث فئات من الكينونات. وعلى الرغم من سهولة هذه الطريقة أيضاً، فإن هذه الطريقة لا تمكِّن من التعرف على الخصائص المشتركة بين الفئات الثلاث من الموظفين. إضافةً إلى ذلك؛ فعلى مستخدمي قاعدة البيانات توخي الدقة في اختيار الفئات المناسبة عند تعاملهم مع قاعدة البيانات.

3- تمثيل الخصائص المشتركة للفئات الثلاث من الموظفين؛ من خلال نوعٍ رئيسيٍّ واحد بمُسمَّى «موظف» (EMPLOYEE) يتفرع منه ثلاثة أنواع فرعية، هي: «موظفو أجر الساعات»

(HOURLY_EMPLOYEE)، و«موظفو الأجر الشهري» (SALARIED_EMPLOYEE)، و«موظفو الأجر وفق عقود استشارية» (CONSULTANT). وتمكِّن هذه الطريقة من التعرُّف على الخصائص المشتركة بين الفئات المختلفة من الموظفين، وفي الوقت نفسه، معرفة الخصائص التي تنفرد فيها كل فئة من فئات الموظفين.

ويمثل الشكل رقم (3-2) النوع الرئيسي للموظفين (EMPLOYEE) والأنواع الفرعية الثلاثة التي تمثل الفئات الثلاث من الموظفين؛ بحيث تم ربط الخصائص المشتركة بين الفئات الثلاث من الموظفين (ومن ضِمنها المميز (أو المعرف)) بالنوع الرئيسي وربط الخصائص التي تتفرد فيها كل فئة من فئات الموظفين بالنوع الفرعي الذي يمثل فئة الموظفين.



شكل رقم (3–2): النوع الرئيسي لكينونة الموظفين وأنواعها الفرعية الثلاثة.

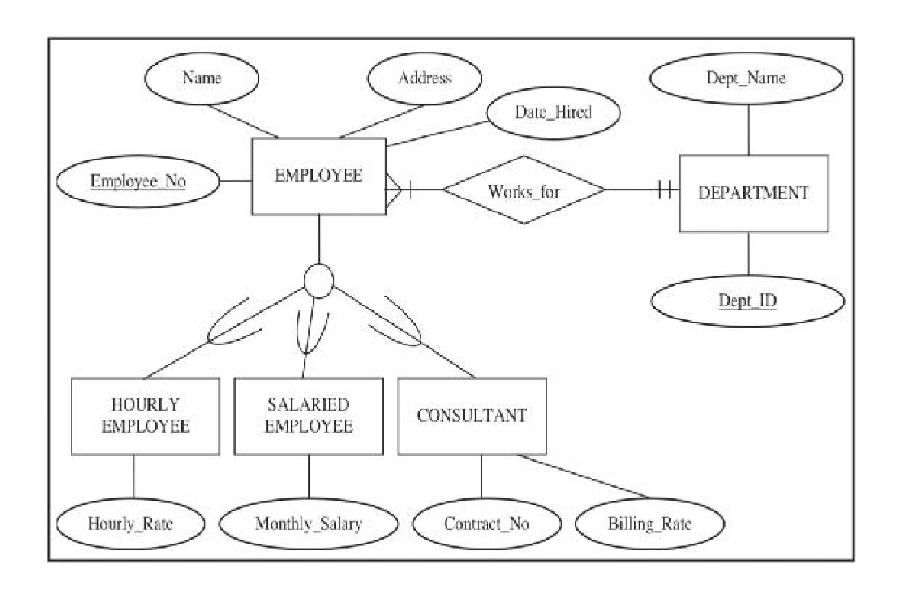
3–1–1–1 توريث الخصائص والعلاقات:

إنَّ أيَّ نوعٍ فرعي يُعَدُّ فئة كينونة قائمة بذاتها. كما أنَّ أية حالة من النوع الفرعي لا بد أن يقابلها نفس الحالة في النوع الرئيسي. ويمكن تصور هذا الوضع على أساس أن كل حالة قد تمَّ

تقسيم خصائصها بين النوع الرئيسي (الذي ترتبط به كافة الخصائص المشتركة لكافة الكينونات بغض النظر عن الأنواع الفرعية التي تتبعها هذه الكينونات)، والنوع الفرعي الذي تتبعه الحالة. فعلى سبيل المثال: لو افترضنا وجود حالة مستشار باسم «صالح الأحمد» (Saleh Al-Ahmed) ضمن النوع الفرعي (CONSULTANT)؛ فإنه لا بد من وجود نفس الشخص ضمن حالات النوع الرئيسي (EMPLOYEE). وبناءً على ذلك؛ فإن أية حالة من حالات أيّ نوع فرعي لا بد أن تتصف، ليس بخصائص النوع الفرعي فحسب، وإنما بخصائص النوع الرئيسي الذي يتبعه النوع الفرعي كذلك.

ويُعرَف الوضع أعلاه باسم توريث الخصائص (Attribute InheritanCe)؛ إذ إن كينونات النوع الفرعي ترث قيماً لكافة خصائص النوع الرئيسي. وبهذه الطريقة يصبح من غير الضروري تكرار الخصائص التي ترتبط بالنوع الرئيسي في الأنواع الفرعية التي تتبعه. فعلى سبيل المثال: إن اسم الموظف؛ هو خاصيةٌ تتبع لفئة كينونة الموظفين ولا تتكرر ضِمن أيٍّ من الأنواع الفرعية التي تتبعها. لذلك؛ فإن اسم المستشار «صالح الأحمد» يُعَدُّ خاصيةً من خصائص فئة كينونة الموظفين وليس من خصائص فئة الكينونة الفرعية (CONSULTANT)؛ غير أن أجر المستشار «صالح الأحمد» خاصية تتبع فئة الكينونة الفرعية (CONSULTANT).

إضافةً إلى توريث الخصائص؛ فإن الأنواع الفرعية ترث العلاقات التي يرتبط بها النوع الرئيسي. فعلى سبيل المثال: لو افترضنا أن أية موظف لا بد أن يعمل في إدارة واحدة فقط من إدارات المنظمة، كما هو مُوضَّح في الشكل رقم (3-3)؛ فإن كافة الأنواع الفرعية لفئة كينونة الموظفين سترث هذه العلاقة بمعنى أن أيَّ مستشار، أو موظف بأجر الساعة، أو موظف بالأجر الشهري لا بدَّ أن يعمل في إدارة واحدة فقط من إدارات المنظمة. ونظراً لكون هذه العلاقة مع فئة كينونة الإدارات علاقة تنطبق على كافة فئات الموظفين، فقد تم ربط هذه العلاقة بالنوع الرئيسي وليس بأيٍّ من الأنواع الفرعية.



شكل رقم (3–3): توريث العلاقات من النوع الرئيسي لأنواعه الفرعية.

وبناءً على ما سبق؛ يمكن طرح السؤال التالي: متى يمكن استخدام علاقة النوع الرئيسي والأنواع الفرعية؟

يعتمد استخدام علاقة النوع الرئيسي والأنواع الفرعية على الحالة التي نحاول نمذجتها. وبشكلٍ عام يقترح وجود أيٍّ من الحالتين التاليتين استخدام علاقة النوع الرئيسي والأنواع الفرعية:

1- وجود خصائص ترتبط ببعض الكينونات؛ ولكن ليس بكافة أنواع الكينونات، مثل: كينونة الموظفين التي تطرقنا إليها أعلاه.

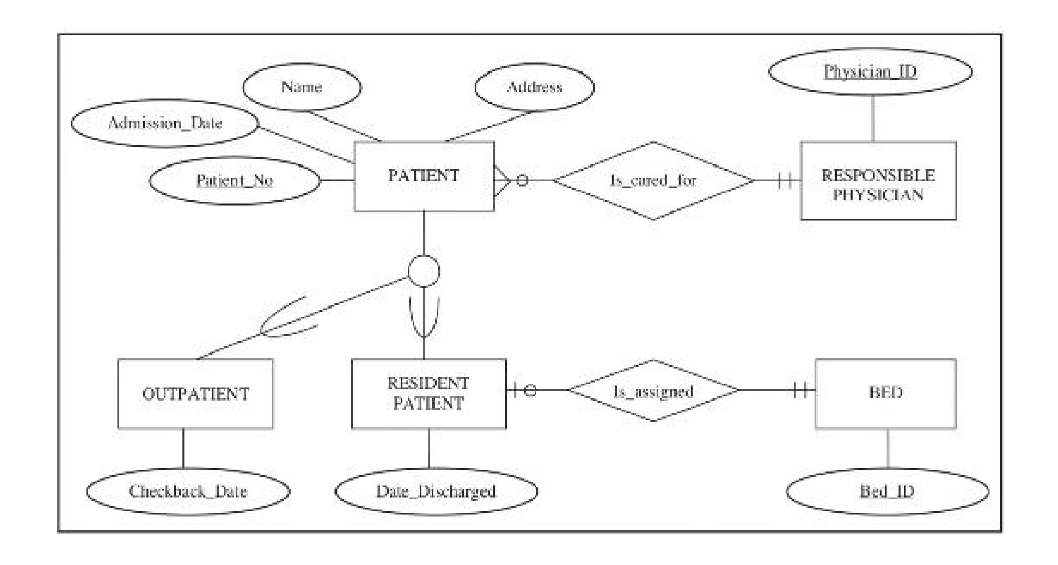
2- ارتباط إحدى فئات الكينونات الفرعية بعلاقةٍ لا ترتبط فيها أيٌّ من فئات الكينونات الفرعية الأخرى.

والشكل رقم (3-4) يحتوي على جزءٍ من نموذج كينونة - علاقة لإحدى المستشفيات يُعَدُّ مثالاً لإيضاح كلتا الحالتين السابقتين؛ فكينونة «المريض» (PATIENT) هي نوعٌ رئيسي يرتبط به نوعان فرعيان، هما: «المرضى المنومون» (RESIDENT_PATIENT) و«مرضى العيادات الخارجية» (OUTPATIENT). ويرتبط بالنوع الرئيسي كافة الخصائص المشتركة لكينونات المرضى، وهي: اسم المريض، عنوان المريض، وتاريخ المراجعة، ورقم المريض (وهو مميز الكينونة). إضافةً إلى ذلك؛ فإن كلَّ مريض يرتبط بعلاقةٍ مع «طبيب مسئول»

(RESPONSIBLE_PHYSICIAN) عن علاج المريض. ولكون كافة المرضى مرتبطين بعلاقات مع أطباء مسئولين عن علاجهم بغض النظر عن كونهم مرضى منومين أو مرضى عيادات خارجية؛ فقد تمَّ ربطُ هذه العلاقة مع الأطباء المسئولين؛ من خلال النوع الرئيسي عوضاً عن أيٍّ من الأنواع الفرعية.

أمَّا بالنسبة للأنواع الفرعية؛ فإن كلاً منها يرتبط بخاصية لا يرتبط بها النوع الآخر. فمرضى العيادات الخارجية يرتبطون بخاصية «تاريخ المراجعة التالية» (CheCkbaCk_Date) التي لا يرتبط بها المرضى المنومون، والمرضى المنومون يرتبطون بخاصية «تاريخ الخروج» (Date_DisCharged) التي لا يرتبط بها مرضى العيادات الخارجية. كما أن المرضى المنومين يرتبطون بعلاقة خاصة بهم وهي علاقة «مسند إلى» (Is_assigned). وهذه العلاقة تربط كلَّ مريض منوم بالسرير الطبي الذي يُسنَد إليه في أثناء فترة بقائه للعلاج في المستشفى. ونظراً لكون هذه العلاقة لا تنطبق على مرضى العيادات الخارجية؛ فإنه قد تم ربطها بالمرضى المنومين فقط. ويعني هذا أن العلاقات الخاصة بالأنواع الفرعية لا تورث للأنواع الفرعية الأخرى ضِمن نفس النوع الرئيسي، أو للنوع الرئيسي نفسه، وإنما تبقى خاصة بالنوع الفرعي الذي يرتبط بها فقط.

ومن خلال عملية توريث الخصائص؛ فإن أيَّ مريض سواء كان منوماً أو مريضاً في العيادات الخارجية سيرث كافة خصائص النوع الرئيسي دون الحاجة إلى تكرارها ضمن خصائصه. فعلى سبيل المثال: لكل مريض سواء كان منوماً أو من مرضى العيادات الخارجية رقمٌ، واسمٌ، وعنوانٌ، وتاريخٌ للمراجعة يرثها من النوع الرئيسي. ومما سبق يتضح أن توريث الخصائص والعلاقات يتم من الأعلى إلى الأسفل، أيْ: من النوع الرئيسي إلى أنواعه الفرعية، وأن عملية التوريث ليست انعكاسية بمعنى أن النوع الرئيسي، أو الأنواع الفرعية الأخرى، لا ترث الخصائص والعلاقات التي يرتبط بها نوعٌ فرعيٌّ ما.



شكل رقم (3–4): مثال لإيضاح الحالات التي يُفضل فيها
استخدام علاقات الأنواع الرئيسية والأنواع الفرعية.

3–1–1–2 توصيف القيود في علاقات الأنواع الرئيسية والأنواع الفرعية:

يُوضِّح هذا الجزء الرموز المستخدَمة لتوصيف القيود في علاقات الأنواع الرئيسية والأنواع الفرعية؛ إذ تمكننا هذه القيود من تمثيل بعض قواعد العمل المهمة عند استخدام نموذج كينونة - علاقة في نمذجة بيانات المنظمة. وأهم نوعين من القيود هما «قيد التخصيص» (SpeCialization Constraint) و«قيد الانفصال» (Disjointness Constraint) (Elmasri and Navathe, 2015).

3–1–1–2–1 قيد التخصيص (**SpeCialization Constraint**):

لقد أشرنا أعلاه إلى أن كلَّ حالة موجودة في أحد الأنواع الفرعية لا بد أن يُوجَد ما يقابلها في النوع الرئيسي. ولكن السؤال هو: هل هذه العملية عكسية؟ بمعنى هل يجب أن تكون كل حالة موجودة في النوع الرئيسي ممثلةً ضِمن إحدى حالات نوع فرعي واحد على الأقل؟

يجيب قيد التخصيص عن هذا التساؤل من خلال قاعدتين، هما: قاعدة التخصيص الكامل وقاعدة التخصيص الجزئي. إن قاعدة التخصيص الكامل تعني أن أيَّ حالة موجودة في النوع الرئيسي لا بد أن تُمثَّل ضِمن واحدة على الأقل من الأنواع الفرعية. أما القاعدة الثانية، قاعدة

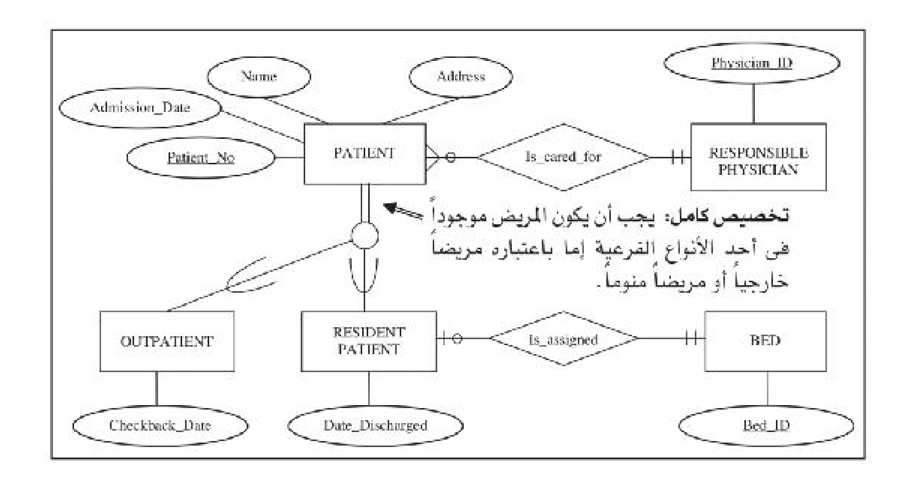
التخصيص الجزئي؛ فتعني أنه من الممكن لحالة مُمثلة ضِمن النوع الرئيسي ألا تُمثَّل ضِمن أيٍّ من الأنواع الفرعية. وفيما يلي إيضاح لكلتا القاعدتين.

يتعلق قيد التخصيص بالعلاقة الرأسية بين نوعٍ رئيسي ما، والأنواع الفرعية المرتبطة به من حيث وجوب أن تُمثَّل كلُّ حالة موجودة ضِمن النوع الرئيسي ضِمن الأنواع الفرعية المرتبطة به، أو عدم وجوب ذلك. فإذا كان من الواجب تمثيل كل حالة ضمن الأنواع الفرعية يُعَدُّ قيد التخصيص كاملاً. أما إذا لم يكن من الواجب ذلك، يُعَدُّ التخصيص جزئياً.

3–1–1–2–1–1 التخصيص الكامل (Total SpeCialization):

يمثِّل الشكل رقم (3-5) المثال السابق المتعلق بمرضى إحدى المستشفيات بعد إيضاح رمز التخصيص عليه. إن قاعدة العمل المعمول بها في هذا المثال هي كما يلي: «إن أيَّ مريضٍ في المستشفى لا بد أن يكون إما مريضاً منوماً أو مريضاً في العيادات الخارجية». ويعني هذا أنه لا يوجد أيُّ نوع من أنواع المرضى سوى الفئتين السابقتين منهم. كما يعني هذا أن قيد التخصيص هو تخصيص كامل؛ فأي حالة لمريض موجود في النوع الرئيسي لا بد أن يُوجَد ما يقابلها في أحد الأنواع الفرعية. ويُمثَّل قيد التخصيص الكامل بخطين مزدوجين يصلان بين النوع الرئيسي وهو «المريض» (PATIENT) ونقطة تفرُّع الأنواع الفرعية (وهي الدائرة).

ويعني هذا التمثيل للتخصيص؛ أنه عندما تتم عملية إضافة مريض جديد لحالات كينونة المرضى لا بد أن يُضاف المريض إما للنوع الفرعي الذي يمثل «المرضى المنومين» (Resident Patients)، أو النوع الفرعي الذي يمثل «مرضى العيادات الخارجية» (Outpatients). وفي حالة إضافة المريض ضمن حالات المرضى المنومين؛ فإنه لا بد أيضاً من إضافة حالة للعلاقة «يُسنَد إلى» (Is_assigned)؛ حتى يتم ربط المريض بسرير معين في المستشفى.

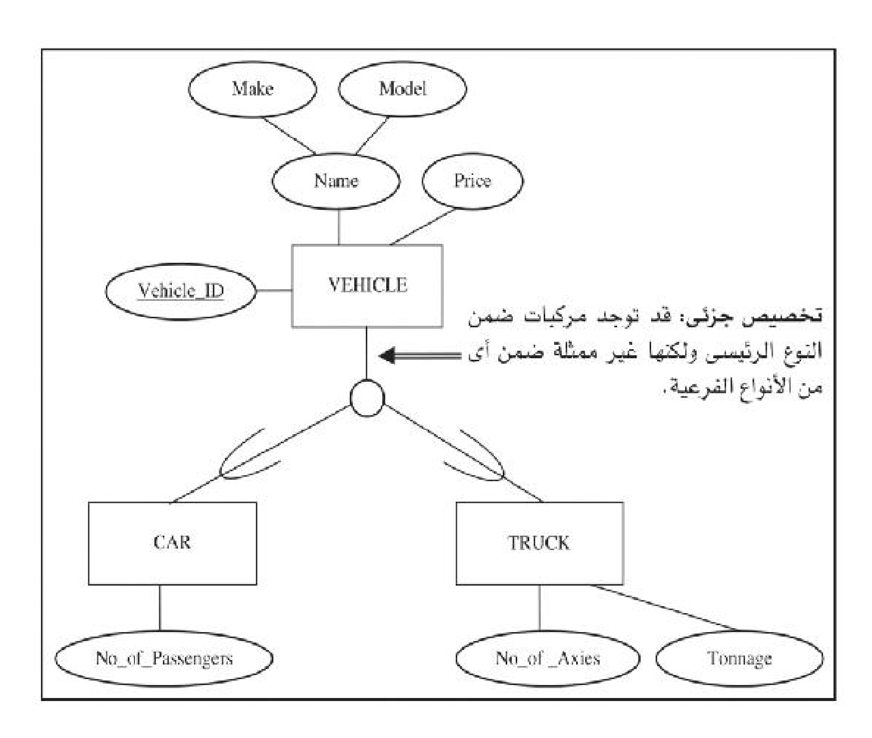


شكل رقم (**3–5**): تمثيل التخصيص الكامل في علاقات الأنواع الرئيسية والأنواع الفرعية.

3–1–1–2–1–2 التخصيص الجزئي (**Partial SpeCialization**):

يُوضِّح الشكل رقم (3-6) مثالاً لنوع رئيسي اسمه «مركبة» (VEHICLE) ونوعين فرعيين يرتبطان به، هما: «سيارة» (CAR) وشاحنة (TRUCK). وكما أوضحنا أعلاه؛ فإن الخصائص المشتركة لكافة المركبات (بغض النظر عن أنواعها الفرعية) قد تم ربطها بالنوع الرئيسي. فرقم لوحة المركبة (VehiCle_ID)، واسمها (Name) (الذي يتكون من صنعها (Make) مثل تويوتا كورولا، وسنة الصنع (Model))، وسعرها (PriCe)؛ هي خصائص مشتركة لكافة أنواع المركبات. أمَّا الخصائص المتعلقة بنوع فرعي معين، مثل: خاصية «عدد الركاب» (No_of_Passengers)، وخاصية «عدد محاور الدفع» (No_of_Axies)، وخاصية «الحمولة بالطن» (Tonnage)؛ فقد تمَّ ربطها بالنوع الفرعي نفسه. ولنفترض في هذا المثال وجود أنواع أخرى من المركبات التي يأتي من ضِمنها «الدراجات النارية» (MOTORCYCLE). ولأن هذا النوع من المركبات لا يُوجَد له نوعٌ فرعي خاص به بمعنى أنَّ الخصائص المشتركة المرتبطة بالنوع الرئيسي تكفي لتمثيله دون وجود خصائص تتعلق به فقط دون غيره من المركبات؛ فإن قيد التخصيص في هذه الحالة يُعَدُّ تخصيصاً جزئياً. فعند إضافة مركبة (VEHICLE) من فئة سيارة ضمن حالات النوع الرئيسي لا بد من إضافة ما يقابلها ضمن حالات النوع الفرعي «سيارة» (CAR)، وعند إضافة مركبة جديدة (VEHICLE) من فئة شاحنة ضِمن النوع الرئيسي لا بد من إضافة ما يقابلها ضمن حالات النوع الفرعي «شاحنة»

(TRUCK)، أما عند إضافة مركبة (VEHICLE) من فئة «دراجة نارية»؛ فإنه يُكتفى بـإضافتها ضِمن حالات النوع الرئيسي فقط. ويمثل التخصيص الجزئي بخط منفرد يصل النوع الرئيسي وهو «مركبة» (VEHICLE) بنقطة تفرع الأنواع الفرعية (وهي الدائرة).



شكل رقم (3–6): تمثيل التخصيص الجزئي في علاقات الأنواع الرئيسية والأنواع الفرعية.

3–1–1–2–2 قيد الانفصال (Disjointness Constraint):

إنَّ قيد التخصيص الذي سبق شرحه أعلاه يتحدث عن العلاقة الرأسية بين النوع الرئيسي وأنواعه الفرعية، بمعنى أنه يقيد حالات النوع الرئيسي؛ من حيث ضرورة وجود كلِّ حالة ضِمن الأنواع الفرعية التي ترتبط بـه من عدم وجودها. أمَّا «قيد الانفصال» (Disjointness Constraint)؛ فيتحدث عن العلاقة الأفقية بين الأنواع الفرعية التي تتبع لنوعٍ رئيسي معين؛ بـمعنى أنه يقيِّد الأنواع الفرعية؛ من حيث إمكانية وجود حالة معينة في أكثر من نوعٍ فرعي واحد في نفس الوقت. وكما هو الحالُ في قيد التخصيص؛ يُوجَد قاعدتان لقيد الانفصال، هما: الانفصال الكامل والانفصال المتداخل. فالانفصال الكامل يعني أن وجود حالةٍ ما من حالات النوع الرئيسي في نوع فرعي معين لا يمكن أن تُوجَد ضمن حالات أنواع فرعية أخرى ترتبط بـنفس النوع

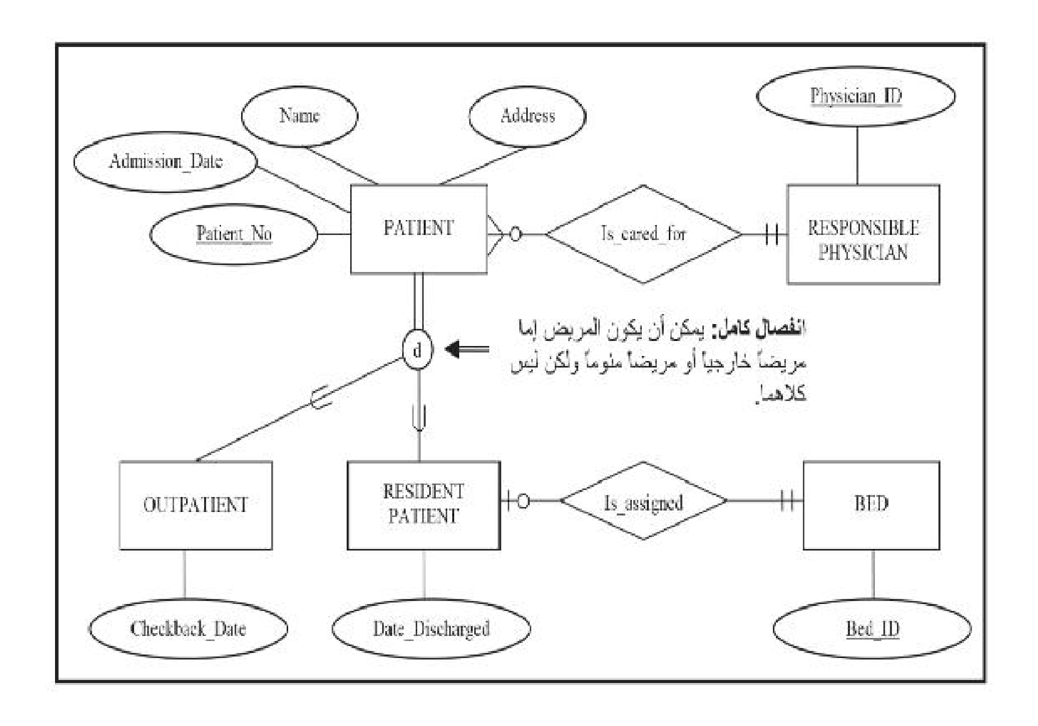
الرئيسي. أمَّا الانفصال المتداخل؛ فيعني أن وجود حالة ما من حالات النوع الرئيسي في نوعٍ فرعي ما من الممكن أن تُوجَد في أنواع فرعية أخرى ترتبط بـنفس النوع الرئيسي. وفيما يلي إيضاحٌ لكلتا القاعدتين.

يتعلَّق قيد الانفصال بالعلاقة الأفقية بين الأنواع الفرعية المرتبطة بنوعٍ رئيسيٍّ ما؛ من حيث إمكانية تمثيل حالة موجودة في نوعٍ فرعي ما ضِمن حالات نوع فرعي آخر مرتبط بـنفس النوع الرئيسي من عدم إمكانية ذلك. فإذا كان من الممكن ذلك؛ أصـبـح قيد الانفصال متداخلاً. أما إن لم يكن بالإمكان ذلك أصبـح قيد الانفصال كاملاً.

3-1-1-2-2-1 الانفصال الكامل (Total Disjoint):

يُبيِّن الشكل رقم (3-7) المثال المتعلِّق بمرضى إحدى المستشفيات بـعد إيضاح رمز الانفصال عليه. إن قاعدة العمل المعمول بـها في هذا المثال هي كما يلي: «إن أيَّ مريضٍ في المستشفى إما أن يكون مريضاً منوماً أو مريضاً في العيادات الخارجية؛ ولكنه لا يمكن أن يكون مريضاً منوماً ومريضاً في العيادات الخارجية في وقتٍ واحد». وتعني قاعدة العمل هذه أن القيد بـين الأنواع الفرعية؛ هو انفصالٌ كاملٌ؛ إذ إنه لا يمكن أن يُوجَد أيُّ مريض ضِمن حالات أكثر من نوع فرعي واحد في وقتٍ واحد. ويُمثّل قيد الانفصال الكامل بالحرف (d)، الذي يمثل الحرف الأول من كلمة «انفصال» (Disjoint)، داخل دائرة التفرع.

ويُلاحَظ في هذا التمثيل أنَّ حالة أيِّ مريضٍ من الممكن أن تكون في أيِّ نوع فرعي، ولكنها لا يمكن أن تُوجَد في أكثر من نوع فرعي في نفس الوقت. فالمريض المنوَّم قد يخرج من المستشفى ويصـبح مراجعاً للعيادات الخارجية، وكذلك هو الحال بالنسبة لمريض العيادات الخارجية الذي قد يصبـح مريضاً منوماً في وقتٍ آخر. ويمكن قراءة القيدين الممثلين في الشكلِ (قيد التخصيص وقيد الانفصال) مجتمعين، كما يلي: يُوجَد تخصيصٌ كاملٌ بين النوع الرئيسي والأنواع الفرعية المرتبطة به؛ لكون أيِّ مريض في المستشفى لا بد أن يمثل ضمن الأنواع الفرعية، ويوجد انفصالٌ كامٌل لكون أيِّ مريض لا يمكن أن يكون موجوداً في أكثر من نوعٍ فرعي واحد في أيِّ وقت.



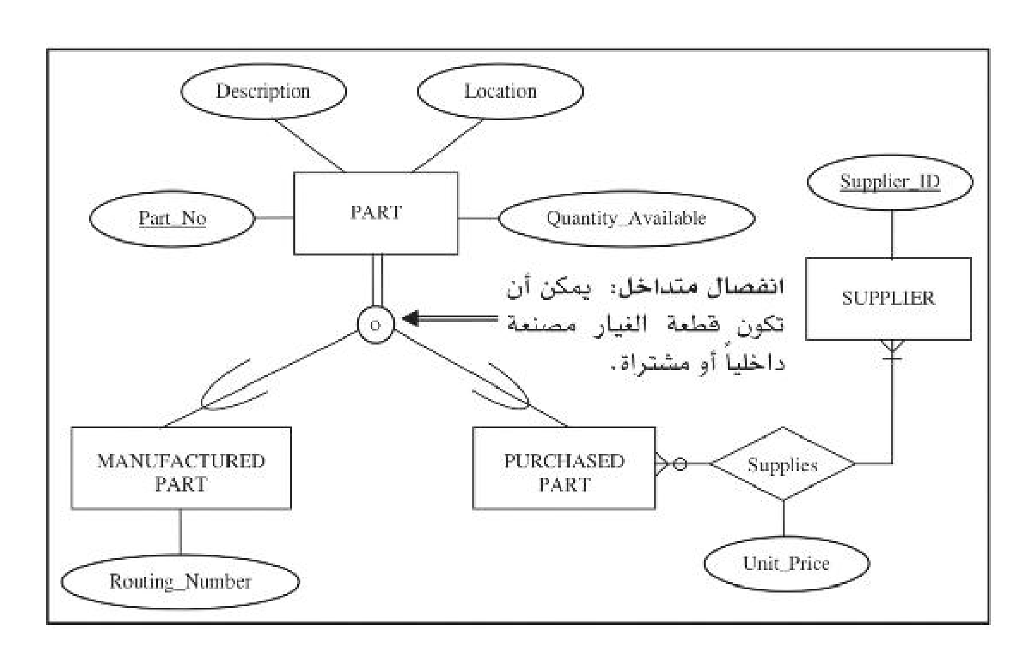
شكل رقم (3–7): تمثيل الانفصال الكامل في علاقات الأنواع الرئيسية والأنواع الفرعية.

3–1–1–2–2–2 الانفصال المتداخل (Overlapping Disjoint):

يُوضِّح الشكل رقم (3-8) فئة كينونة «قطعة غيار» (PART) يرتبط بها نوعان فرعيان، هما: قطع الغيار المصنَّعة داخلياً (في نفس المنظمة) بمُسمَّى (MANUFACTURED_PART)، وقطع الغيار المشتراة (PURCHASED_PART). في هذا المثال تمَّ ربط الخصائص المشتركة لقطع الغيار بالنوع الرئيسي. وهذه الخصائص، هي: رقم قطعة الغيار (Part_No) الذي يُعَدُّ معرفاً لقطع الغيار، ووصف لقطعة الغيار (DesCription)، ومكان وجود قطعة الغيار (LoCation)، والعدد المتوفِّر منها (Available_Quantity). فعلى سبيل المثال: قد يتوفر من قطعة الغيار رقم (500)، وهي مصابيح خلفية يسرى لسيارة فورد توراس 2004، (15) قطعة موجودة في الرف رقم (4) من الممر رقم (5) في مستودع المنظمة. ولنفترض أنه تمَّ تصنيع (10) قطع من قطعة الغيار هذه محلياً، في حين تمَّ توريد (5) قطع من مورد ما. في هذه الحالة تمثِّل قطعة الغيار هذه ضِمن حالات النوع الفرعي (MANUFACTURED_PART) وفي نفس الوقت ضِمن حالات النوع الفرعي (PURCHASED_PART). ويعني هذا التمثيل أن قيد الانفصال متداخلٌ؛ لكون قطع الغيار قد تُوجَد في أكثر من نوعٍ فرعي واحد في وقتٍ واحد. ويُلاحَظ في هذا التمثيل أن النوع الفرعي

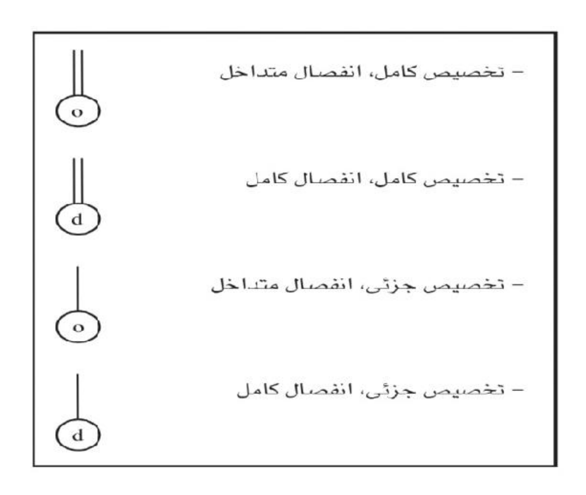
المتعلق بالقطع المصنَّعة محلياً له خاصية تميِّزه عن النوع الفرعي المتعلق بقطع الغيار المشتراة من مورد ما، وأن قطع الغيار المشتراة ترتبط بعلاقة مع كينونة «المورد» (SUPPLIER) التي لا يرتبط فيها النوع الفرعي الآخر. كما يُلاحَظ في هذا التمثيل أنه لا يتتبع كل قطعة على حِدة؛ ولكنه يتتبع مجموعةً من القِطع من نفس النوع، ولها نفس الرقم كمجموعة واحدة. أما إذا أردنا تمثيل كل قطعة على حِدة؛ فإنه يجب إدخال الرقم التسلسلي لكلِّ قطعة (Serial Number) وتكون الكمية المتوفرة منها إما (1) أو (0) حسب وجودها في المستودع، مع الإبقاء على بقية مكوِّنات التمثيل في النموذج كما هي.

ويمثل قيد الانفصال المتداخل بالحرف (O)، الذي يمثل الحرف الأول من كلمة «متداخل» (Overlap)، داخل دائرة التفرع. ويمكن قراءة القيدين الممثلين في الشكل (قيد التخصيص وقيد الانفصال) مجتمعين، كما يلي: يُوجَد تخصيص كاملٌ بين النوع الرئيسي والأنواع الفرعية المرتبطة به بمعنى أن أية قطعة غيار؛ يجب أن تُوجَد ضِمن الأنواع الفرعية، ويُوجَد انفصالٌ متداخل؛ لكون بعض قطع الغيار قد توجد ضِمن قطع الغيار المشتراة وفي الوقت نفسه ضِمن قطع الغيار المصنَّعة محلياً.



شكل رقم (3–8): تمثيل الانفصال المتداخل في علاقات الأنواع الرئيسية والأنواع الفرعية.

يحتوي الشكل رقم (3-9) على كافة التوليفات الأربع المحتملة للقيدين السابقين (قيد التخصيص، وقيد الانفصال) التي يمكِن استخدام المناسب منها حسب قواعد العمل المعمول بها في المنظمة؛ وذلك عند استخدام علاقة النوع الرئيسي والأنواع الفرعية.



شكل رقم (3–9): التوليفات الأربع المحتملة لقيد التخصيص وقيد الانفصال.

3–1–1–3 تعريف مميّز للأنواع الفرعية (Defining Subtype Identifiers):

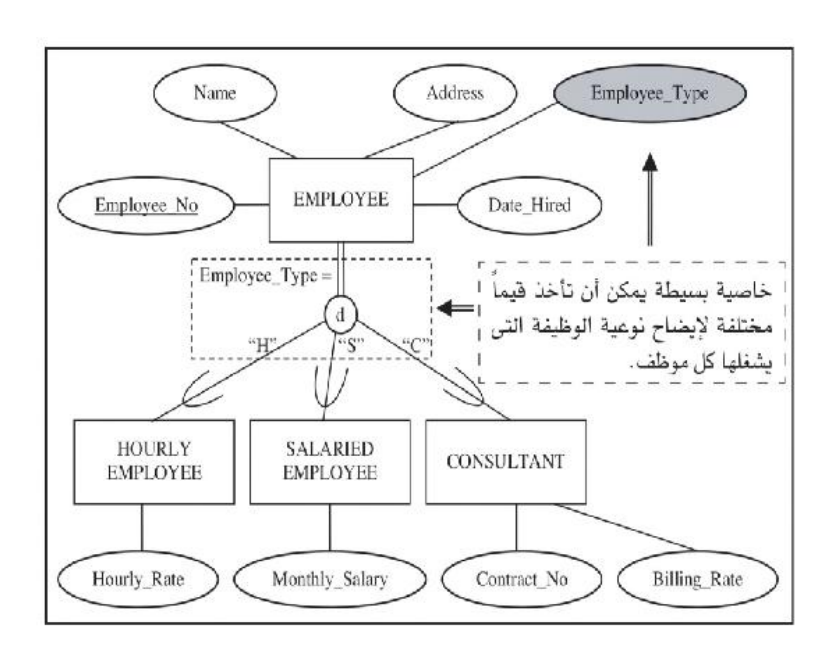
عند إضافة حالة جديدة ضِمن حالات نوعٍ رئيسيٍّ معينٍ؛ فإننا نحتاجُ إلى معرفة النوع الفرعي (أو الأنواع الفرعية) التي يجب إضافة هذه الحالة إليها إذا كان لا بد من إضافة الحالة إلى أحد الأنواع الفرعية. ولأننا قد سبق أن شرحنا قواعد الإضافة لكلٍّ من قيد التخصيص وقيد الانفصال؛ فإننا نقدِّم هنا طريقةً مبسطةً تعتمد على مميِّز للأنواع الفرعية لتطبيق قواعد الإضافة هذه (Hoffer et al, 2018). ومميِّز الأنواع الفرعية ما هو إلا خاصية تتمُّ إضافتها للنوع الرئيسي؛ بحيث تحدِّد النوع الفرعي (أو الأنواع الفرعية) التي يجب إضافة الحالة إليها.

3–1–1–3–1 الانفصال الكامل (Total Disjoint):

يمثِّل الشكل رقم (3-10) علاقة النوع الرئيسي «موظف» بالأنواع الفرعية الثلاثة التي ترتبط به والتي سبق شرحها أعلاه؛ إذ يُوجَد تخصيصٌ كاملٌ بين النوع الرئيسي والأنواع الفرعية، بمعنى أن أيَّ موظفٍ موجود ضمن حالات النوع الرئيسي لا بد أن تُوجَد له حالةٌ في الأنواع

الفرعية. كما يُوجَد انفصالٌ كاملٌ بين الأنواع الفرعية، بمعنى أن أيَّ موظفٍ لا يمكن أن تُمثَّل حالته في أكثر من نوعٍ فرعيٍّ واحد.

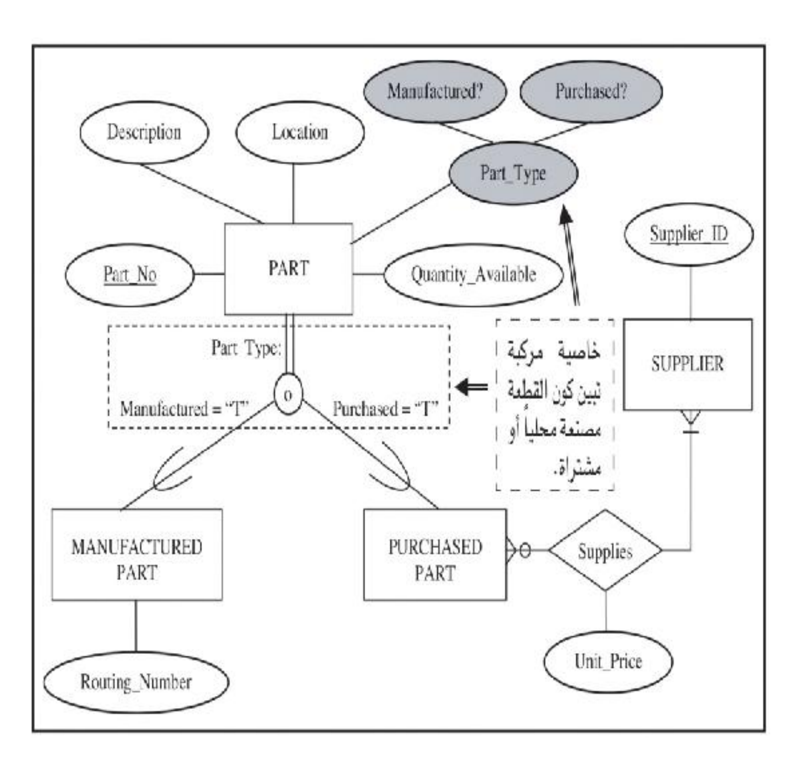
ويُوضِّح الشكل رقم (3-10) مميز الأنواع الفرعية الذي حُدِّد باللون الداكن بمُسمَّى «نوع الموظف» (Employee_Type). وعند إضافة موظف جديد؛ فإنه يجب إدخال قيمة لهذا المميز؛ بحيث تكون قيمته إما (H) للدلالة على أن الموظف يعمل بأجر الساعات، أو (C) للدلالة على أن الموظف يعمل مستشاراً وفق عقدٍ استشاري، أو (S) للدلالة على أن الموظف يعمل بالأجر الشهري. وبناءً على القيمة المُدخَلة لهذه الخاصية؛ تتمُّ إضافة حالة للموظف الجديد ضِمن حالات النوع الفرعي المناسب. كما يُستخدَم في هذا التمثيل اسم المميّز بين النوع الرئيسي ونقطة التفرُّع كشرطٍ يجب أن تساوي قيمته إحدى القيم المدونة على الخطوط الواصلة بين نقطة التفرُّع والأنواع الفرعية. فعلى سبيل المثال: عند إضافة موظف قيمة خاصية «نوع الموظف» (Employee_Type) هي «مستشار» (C)؛ فإن هذا يعني أن شرط الإضافة سيكون (Employee_Type = "C")؛ مما يؤدي إلى إضافة حالة لهذا الموظف ضمن النوع الفرعي (CONSULTANT).



شكل رقم (3–10): تعريف مميز الأنواع الفرعية في حالة الانفصال الكامل.

3–1–1–3–2 الانفصال المتداخل (Overlapping Disjoint):

يمثِّل الشكل رقم (3-11) علاقة النوع الرئيسي «قطعة غيار» بالنوعين الفرعيين اللذين يرتبطان به، وقد سبق شرحها أعلاه؛ إذ يُوجَد تخصيصٌ كاملٌ بين النوع الرئيسي والأنواع الفرعية، بمعنى أن أيَّ قطعة غيار موجودة ضِمن حالات النوع الرئيسي لا بـد أن تُوجَد لها حالةٌ في الأنواع الفرعية. كما يُوجَد انفصالٌ متداخلٌ بين الأنواع الفرعية، بـمعنى أن قطعة الغيار قد تُوجَد ضِمن قطع الغيار المصنعة محلياً وفي الوقت نفسه ضمن قطع الغيار المشتراة.



شكل رقم (3–11): تعريف مميّز الأنواع الفرعية في حالة الانفصال المتداخل.

ويوضِّح الشكل رقم (3-11) مميز الأنواع الفرعية الذي حُدِّد باللون الداكن بمُسمَّى «نوع قطعة الغيار» (Part_Type)، والذي يختلف بشكلٍ بسيطٍ عن التمثيل السابق؛ إذ مثل المميز كخاصية مركبة عوضاً عن تمثيله كخاصية بـسيطة. وبهذه الطريقة نستطيع تمثيل التوليفات المناسبة لكلِّ قطعة غيار؛ من حيث كونها مصنعةً محلياً أو مشتراة. وتأخذ كلُّ خاصية بسيطة من مكونات الخاصية المركبة قيمة منطقية واحدة، هي: إما صح (True) أو خطأ (False). فعندما يتمُّ إدخال قطعة غيار جديدة ضمن حالات النوع الرئيسي وتكون القطعة مُصنَّعة محلياً ومشتراة في نفس الوقت تكون قيم الخاصيتين البسيطتين، كما يلي: (ManufaCtured = True و PurChased = True). أما إذا كانت القطعة مصنعة محلياً فقط؛ فتكون قيم الخاصيتين

البسيطتين، كما يلي: (ManufaCtured = True و PurChased = False). أمَّا إذا كانت القطعة مشتراة فقط فتكون قيم الخاصيتين البسيطتين، كما يلي:

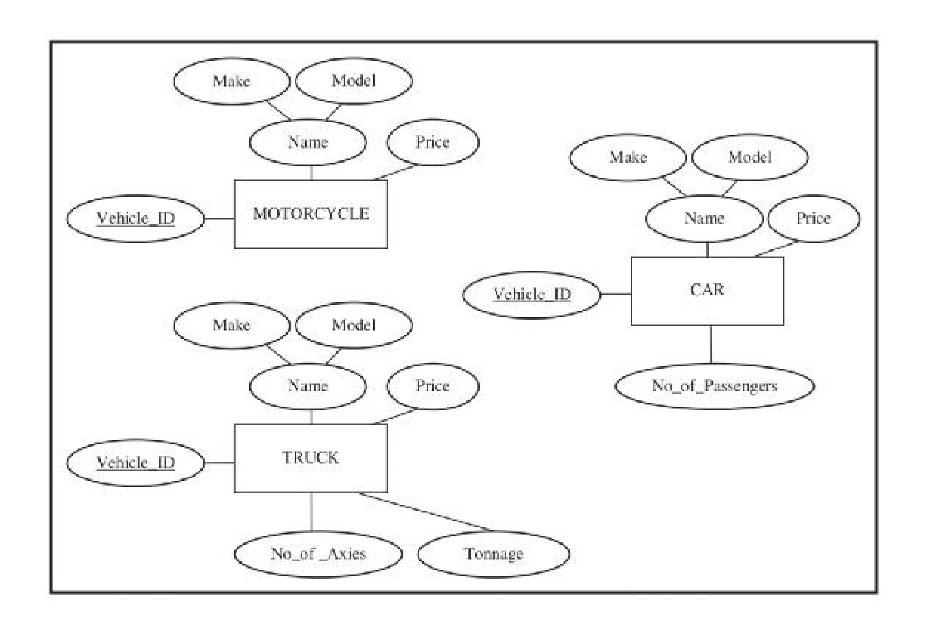
(ManufaCtured = False و PurChased = True)

3–1–1–4 التعميم والتخصيص (Generalization and SpeCialization):

لا شك أن علاقة الأنواع الرئيسية والأنواع الفرعية في نموذج البيانات «كينونة - علاقة» المطور؛ مبدأ جيد يمكِّننا من وصف العلاقات بين الكينونات الموجودة في المنظمة بشكلٍ أكثر دقة. غير أن البيانات تختلف من منظمةٍ إلى أخرى، وقد يستعصي على الشخص الذي سيقوم بنمذجة البيانات التعرُّف على الكينونات التي يمكِن أن ترتبط بعلاقة نوع رئيسي وأنواع فرعية منذ البدء في تصميم نموذج البيانات. لهذا يمكِن استخدام عملية التعميم أو التخصيص لاستكشاف مثل هذه العلاقات.

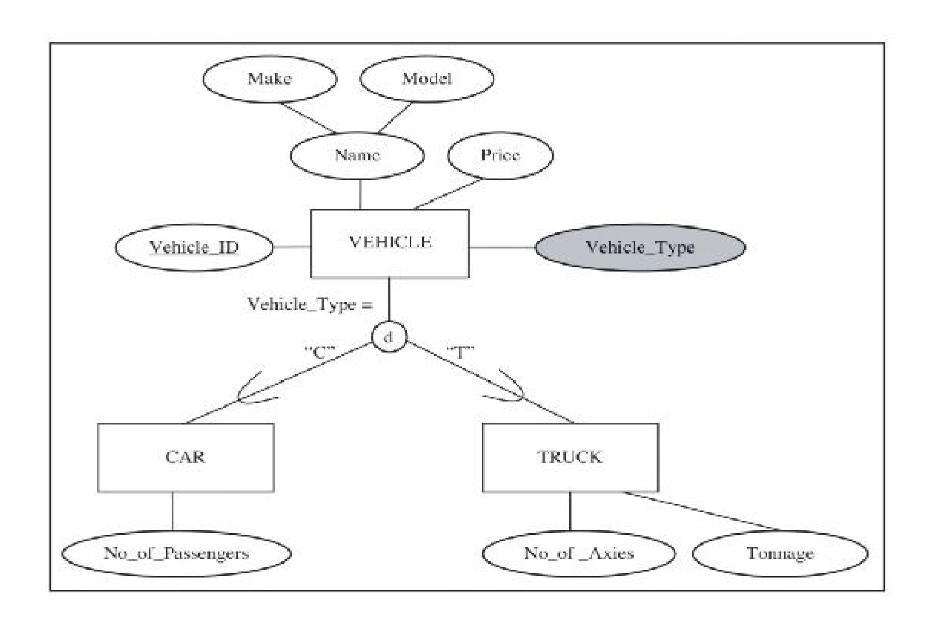
3–1–1–4–1 التعميم (Generalization):

إن عملية التعميم في نمذجة البيانات؛ هي عمليةٌ تتمُّ من الأسفل إلى الأعلى بمعنى أنها عملية تعريف لفئةِ كينونة أعمّ من مجموعة من فئات الكينونات المخصَّصة. فعلى سبيل المثال: يُوضِّح الشكل رقم (3-12) ثلاثة أنواع من الكينونات، هي: «السيارة» (CAR)، و«الشاحنة» (TRUCK)، و«الدراجة النارية» (MOTORCYCLE). وعلى الرغم من أنه يمكن تمثيل هذه الكينونات الثلاث ضِمن نموذج البيانات كينونة - علاقة كما هي؛ فإنه بنظرة فاحصة يتضحُ أن الكينونات الثلاث لديها خصائص مشتركة، وهي: الاسم، والسعر، ورقم المركبة. ويقترح مثل هذا التشابه في بعض الخصائص بين فئات الكينونات المختلفة على مُنمذج البيانات التفكير في إمكانية تعريف فئة كينونة أعم.



شكل رقم (3–12): التعرُّف على الأنواع الرئيسية والأنواع الفرعية من خلال عملية التعميم.

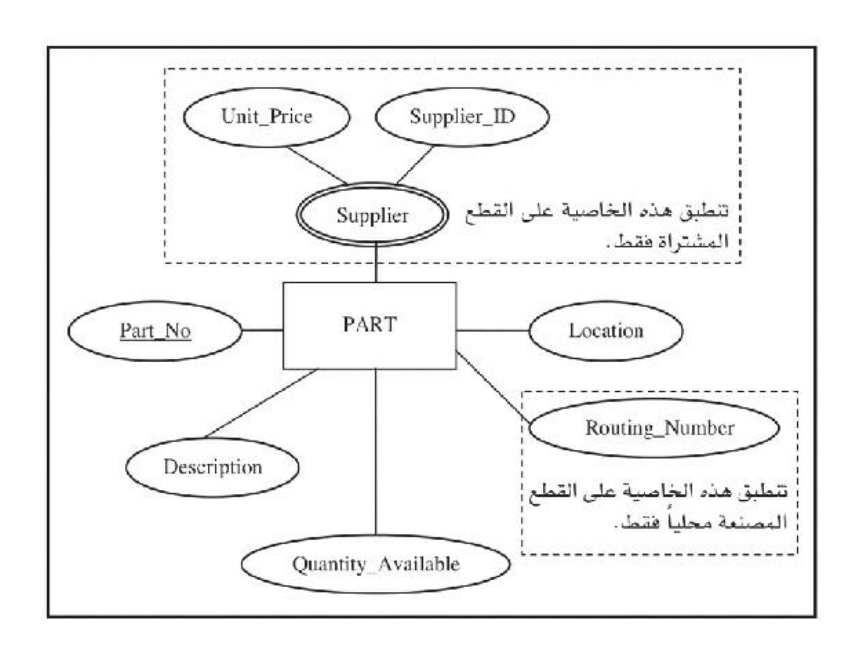
ونظراً لهذا التشابه؛ فإنه يمكن تعريف فئة كينونة أعم وهي كينونة «المركبة» (VEHICLE). ويوضِّح الشكل رقم (3-13) فئة الكينونة الجديدة إضافةً لعلاقة النوع الرئيسي والأنواع الفرعية. ويُلاحَظ في الشكل نفسه وجود نوعين فرعيين فقط عوضاً عن ثلاثة؛ وذلك لكون كينونة السيارة وكينونة الشاحنة لهما خصائص تميِّزهما عن بعضهما، وهذه الخصائص غير موجودة ضِمن الخصائص العامة لكافة الكينونات والمرتبطة بالنوع الرئيسي، أما فئة كينونة الدراجة النارية؛ فجميع خصائصها خصائص عامة ومن ثم لا داعي لتمثيلها بـوصفها نوعاً فرعياً. كما يتمُّ إضافة القيود بين النوع الرئيسي والأنواع الفرعية، وهي في هذا المثال تخصيصٌ جزئي لكون الدراجات النارية لا تظهر ضِمن أيٍّ من الأنواع الفرعية، وانفصال كامل؛ لأن أيَّ مركبة لا يمكن أن تكون ضِمن حالات فئة كينونة السيارة وفي الوقت نفسه ضِمن فئة كينونة الشاحنة. إضافةً لذلك؛ يتم تعريف خاصية المميز للأنواع الفرعية وربطها بالنوع الرئيسي.



شكل رقم (3–13): نمذجة الأنواع الرئيسية والأنواع الفرعية بعد التعرف عليها من خلال عملية التعميم.

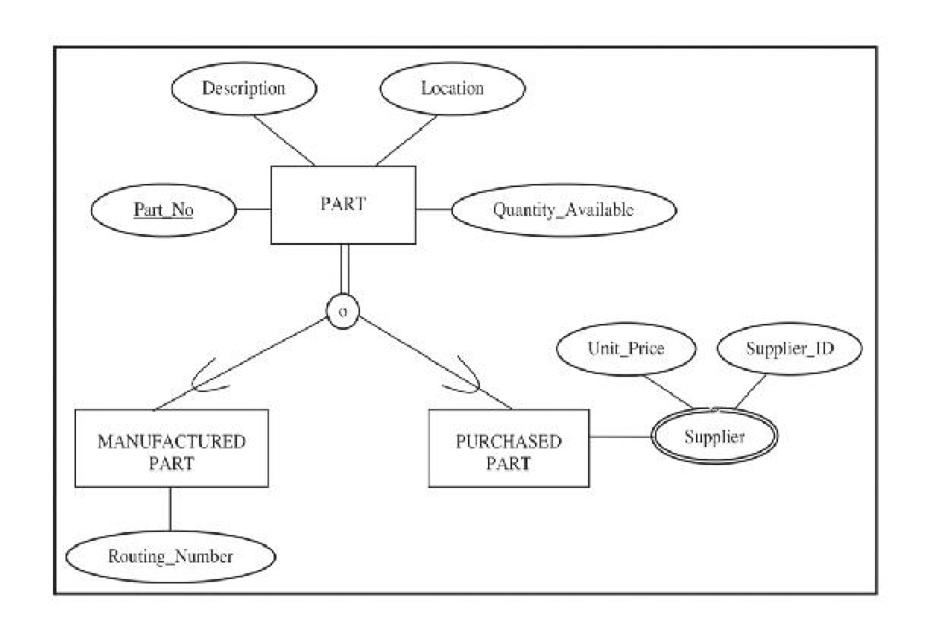
3–1–1–4–2 التخصيص (SpeCialization):

إنَّ عملية التخصيص هي عكس عملية التعميم؛ إذ تتمُّ من الأعلى إلى الأسفل بمعنى أنها عملية تعريف لفئات كينونات مُخصَّصة من فئة كينونة أعم. فعلى سبيل المثال: يُوضِّح الشكل رقم (3-14) كينونة «قطعة غيار» (PART) التي يمكن أن تكون «مشتراة» أو «مُصنَّعة محلياً» (أو كلتيهما معاً في وقتٍ واحد). ويُلاحَظ في هذا التمثيل وجود خاصية متعددة القيم، وهي خاصية «المورد» (Supplier) التي تتكون من الخاصية البسيطة «رقم المورد» (Supplier_ID) وسعر القطعة، مع ملاحظة أن سعر القطعة قد يتغير من مورد إلى آخر. وترتبط هذه الخاصية بقطع الغيار المشتراة فقط. كما يُلاحَظ في هذا التمثيل وجود الخاصية البسيطة «رقم المصدر» (Routing_Number) التي ترتبط بالقطع المصنَّعة محلياً فقط. وفي حالة كون القطع مُصنَّعة محلياً ومُشتراة في نفس الوقت؛ فسيكون للقطعة قِيمٌ ضِمن كلتا الخاصيتين (المورد، ورقم المصدر). ولكون خصائص قطع الغيار قد تختلف حسب طبيعة كون قطعة الغيار مشتراةً أو مصنعةً محلياً؛ فإن مثل هذا التمثيل يقترح على مُنمذِج البيانات التفكير في إمكانية تعريف فئات خاصة لكلِّ نوع من أنواع قطع الغيار.



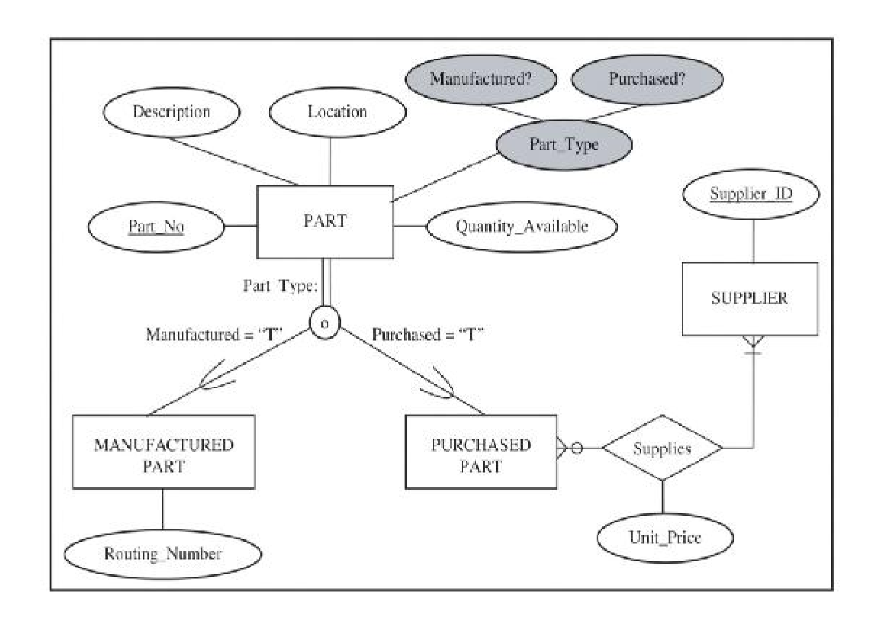
شكل رقم (3–14): التعرُّف على الأنواع الرئيسية والأنواع الفرعية من خلال عملية التخصيص.

ونظراً لوجود اختلاف في بعض خصائص كلا النوعين من أنواع قطع الغيار؛ فإنه يمكن تعريف فئات كينونات فرعية مُخصَّصة، وهي: كينونة «القطع المصنَّعة محلياً» (MANUFACTURED_PART)، وكينونة «القطع المشتراة» (PURCHASED_PART). وترتبط كلتا الكينونتين بالنوع الرئيسي من خلال علاقة النوع الرئيسي والأنواع الفرعية، كما يرتبط كلُّ نوعٍ فرعيٍّ بالخصائص الخاصة به في حين يرتبط النوع الرئيسي بالخصائص العامة لكافة أنواع الكينونات. ونظراً لكون كلِّ قطعة غيار يجب أن تكون إما مصنَّعة محلياً أو مشتراة؛ فإن هذا يعني وَضْع قيد تخصيصٍ كاملٍ. أمَّا قيد الانفصال؛ فهو انفصالٌ متداخلٌ؛ وذلك لكون بعض قطع الغيار قد تكون مصنعةً محلياً ومشتراةً في وقت واحد؛ مما يعني أن بعض الحالات الموجودة في النوع الرئيسي قد تُوجَد في كلا النوعين الفرعيين. وبناءً على ذلك يصبح النموذج السابق كما هو مُوضَّح في الشكل رقم (3-15).



شكل رقم (3–15): نمذجة الأنواع الرئيسية والأنواع الفرعية
بعد التعرُّف عليها من خلال عملية التخصيص.

وبملاحظة كون الخاصية المرتبطة بالقطع المشتراة؛ أنها خاصيةٌ متعددةُ القيم وفي الوقت نفسه تحتوي على قيمة بسيطة تُعَدُّ مميزاً للخاصية وهي «رقم المورد» (Supplier_ID)، فإن هذه الخاصية يمكِن تحويلها إلى كينونة قائمة بذاتها ترتبط بالنوع الفرعي من خلال علاقة «يورد» (Supplies). كما ترتبط بالعلاقة نفسها خاصية «سعر الوحدة» (Unit_PriCe)؛ وذلك لأن سعر الوحدة ليست خاصية من خصائص المورد، كما أنها ليست خاصيةً من خصائص قطعة الغيار، وإنما هي خاصيةٌ للعلاقة نفسها بمعنى أن قيمة هذه الخاصية تختلف باختلاف المورد واختلاف قطعة الغيار. فعلى سبيل المثال: قد يكون لإحدى قطع الغيار المشتراة أكثر من سعر للوحدة حسب المورِّدين الذين قاموا بتوريد القطعة. لذلك لا يمكن ربط هذه الخاصية بأيٍّ من الكينونتين، وإنما يتم ربطها بالعلاقة التي تربط الكينونتين ببعضهما. وبذلك يصبح الشكل النهائي لعملية التخصيص كما هو مُوضَّح في الشكل رقم (3-16)، مع ملاحظة إضافة مميز الأنواع الفرعية «نوع القطعة» (Part_Type) ضِمن خصائص النوع الرئيسي.

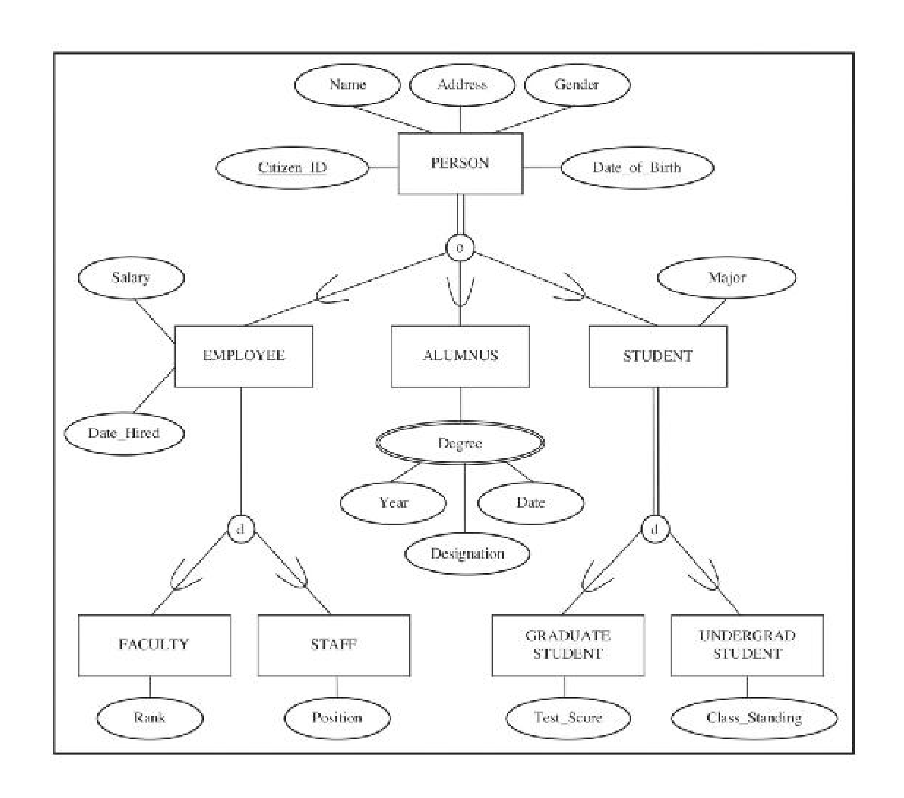


شكل رقم (3–16): تحديد العلاقات الخاصة بالأنواع الفرعية في علاقات الأنواع الرئيسية والأنواع الفرعية.

مما سبق تتضحُ أهمية عمليتي التعميم والتخصيص؛ إذ إنهما تمكِّنان من تفحُّص النموذج المبدئي للبيانات، ومن ثم إمكانية التعرُّف على الأنواع الرئيسية والأنواع الفرعية؛ مما يسهم في تطوير النموذج؛ بحيث يعكس علاقات البيانات والقيود المفروضة عليها بشكلٍ أكثر دقة.

3–1–1–5 هرميات الأنواع الرئيسية والأنواع الفرعية (Supertype/Subtype HierarChies):

عند وجود علاقة نوع رئيسي بأنواع فرعية؛ فإنه من الممكن أن يرتبط كلٌّ نوع من الأنواع الفرعية بأنواع فرعية خاصة به؛ وبذلك تتشكل هرمية من الأنواع الرئيسية والأنواع الفرعية؛ بحيث يصبح النوع الفرعي الذي ترتبط به أنواع فرعية خاصة به نوعاً رئيسياً للأنواع الفرعية المرتبطة به (Elmasri and Navathe, 2015). ويوضِّح الشكل رقم (3-17) أحد الأمثلة الشائعة الذي يمثل بيانات الموارد البشرية في إحدى الجامعات والعلاقات فيما بينها، كما تمَّ إيضاح المفاهيم والرموز التي تم شرحها حتى الآن عليه.



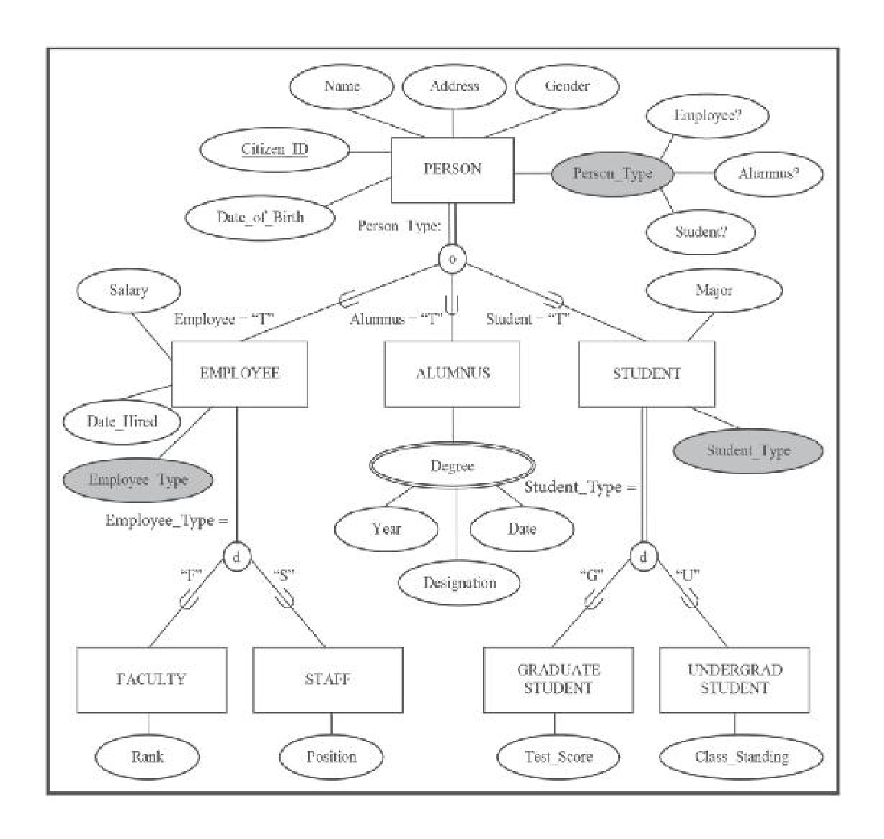
شكل رقم (3–17): أحد الأمثلة الشائعة لهرميات الأنواع الفرعية والأنواع الرئيسية.

لقد تمَّ في الشكل رقم (3-17) تعريف كينونة نوع رئيسي، وهي كينونة «الأشخاص» (PERSON) التي ترتبط فيها كافة الخصائص المشتركة للأشخاص التابعين للجامعة من موظفين وطلبة وخريجين. وهذه الخصائص، هي: رمز «السجل المدني» (Citizen_ID) الذي يمثل مميز الكينونة، و«الاسم» (Name)، و«العنوان» (Address)، و«الجنس» (Gender)، و«تاريخ الميلاد» (Date_of_Birth). بعد ذلك تم تخصيص ثلاثة أنواع فرعية، هي: فئة «الموظفين» (EMPLOYEE)، وفئة «الخريجين» (ALUMNUS)، وفئة «الطلبة» (STUDENT). كما تمَّ ربط كلِّ نوعٍ فرعي بالخصائص المتعلقة به فقط؛ إذ تمَّ ربط النوع الفرعي الذي يمثل الموظفين بخاصية «الراتب» (Salary) وخاصية «تاريخ التعيين» (Date_Hired) التي تمثل خصائص عامة لكافة فئات الموظفين. كما تمَّ ربط النوع الفرعي الذي يمثل الخريجين بخاصية متعددة القيم بمُسمَّى «الدرجة العلمية» (Degree)؛ وذلك لكون الخريج قد يكون حاصلاً على أكثر من درجة علمية من نفس الجامعة. كما تمَّ توصيف الدرجة العلمية المتعددة القِيم على أساس أنها مركبة أيضاً؛ إذ تتكون من الخصائص البسيطة التالية: خاصية «سنة الحصول على الدرجة العلمية» (Year)، وخاصية «تاريخ الحصول على الدرجة العلمية»

(Date)، وخاصية «نوع الدرجة العلمية» (Designation). أمَّا بالنسبة للنوع الفرعي الذي يمثل الطلبة؛ فقد تمَّ ربطه بخاصية «التخصص» (Major). ونظراً لكون أيِّ شخصٍ في الجامعة يجب أن يكون من ضِمن أحد الأنواع الفرعية الثلاثة؛ فقد تمَّ وَضْع قيد تخصيص كامل. أما قيد الانفصال؛ فقد تم وضعه على أساس أنه انفصال متداخل؛ وذلك لكون الخريج قد يكون موظفاً في الجامعة أيضاً، كما أن الموظف قد يكون طالباً في الجامعة أيضاً.

بعد ذلك تمَّ النظرُ في كلِّ نوعٍ فرعيٍّ على حِدَة؛ لمعرفة إمكانية تخصيص أنواع فرعية منه؛ إذ لُوحِظ أن الموظف قد يكون «عضواً لهيئة التدريس» (FACULTY) أو أحد أفراد «الطاقم المساعد في عملية التدريس» (بما في ذلك من إداريين وسكرتارية) (STAFF). وبناءً على ذلك؛ تمَّ تخصيصُ فئتين من النوع الفرعي «موظف»، وتمَّ رَبط كلِّ نوع فرعي بالخصائص التي ترتبط بـه فقط. ويُلاحَظ أن قيد التخصيص؛ هو تخصيصٌ جزئي مما يعني وجود موظفين آخرين لا ينتمون لأيٍّ من النوعين الذين تمَّ تخصيصهما، كما يُلاحَظ أن قيد الانفصال هو انفصالٌ كاملٌ؛ مما يعني أنَّ عضو هيئة التدريس لا يمكِن أن يكون من ضِمن الطاقم المساعد في العملية التدريسية أو العكس. بعد ذلك تمَّ النظر في النوع الفرعي الذي يمثل الطلبة؛ إذ لُوحِظ وجود فئتين من الطلبة، وهما: فئة «طلبة الدراسات العليا» (GRADUATE_STUDENT)، وفئة طلبة «دراسات البكالوريوس» (UNDERGRADUATE_STUDENT)، وتمَّ ربط كلِّ فئة بالخصائص التي ترتبط بـها فقط. ولأن أيَّ طالب لا بد أن يكون إما طالباً في الدراسات العليا أو طالباً في مرحلة البكالوريوس؛ فإنه قد تمَّ تمثيل قيد التخصيص على أنه تخصيصٌ كاملٌ، كما تمَّ تمثيل قيد الانفصال على أنه تخصيصٌ كاملٌ لكون الطالب لا بد أن يكون إما طالباً في مرحلة البكالوريوس، أو طالباً في مرحلة الدراسات العليا ولكن ليس في المرحلتين في نفس الوقت.

تجدر الملاحظة هنا أن أيَّ نوعٍ فرعيٍّ في هرمية الأنواع الرئيسية والأنواع الفرعية يرث كافة الخصائص والعلاقات للأنواع التي تعلوه في الهرمية. كما يتمُّ إضافة خاصية مميز الأنواع الفرعية لكلِّ كينونة يتمُّ تخصيصها، وليس فقط للنوع الرئيسي الذي في أعلى الهرمية كما يوضحه الشكل رقم (3-18).



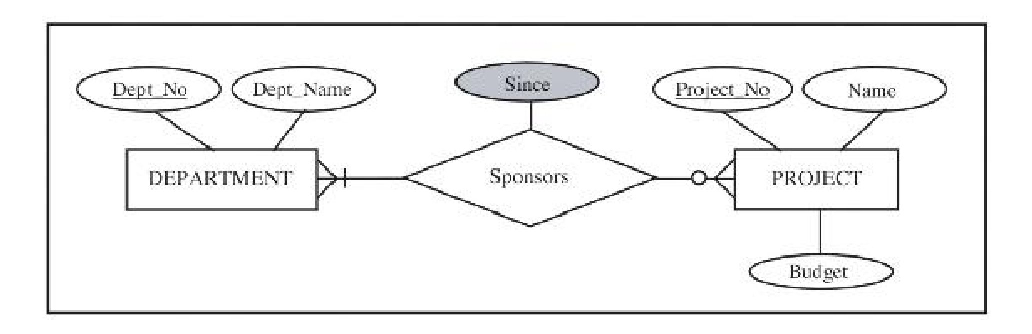
شكل رقم (3–18): إضافة خاصية مميز الأنواع الفرعية في هرميات الأنواع الفرعية والأنواع الرئيسية.

ويُوضِّح المثال السابق أيضاً منهجية التخصيص؛ إذ تمَّ البدء في هذا المثال بالنوع العام وهو «الأشخاص»، وتمَّ تخصيصه شيئاً فشيئاً حسب الخصائص التي تميِّز كلَّ نوع فرعي عن بقية الأنواع الفرعية حتى الوصول للهرمية الكاملة التي تمثل كافة الموارد البشرية في الجامعة.

3–1–1–6 التجميع (Aggregation):

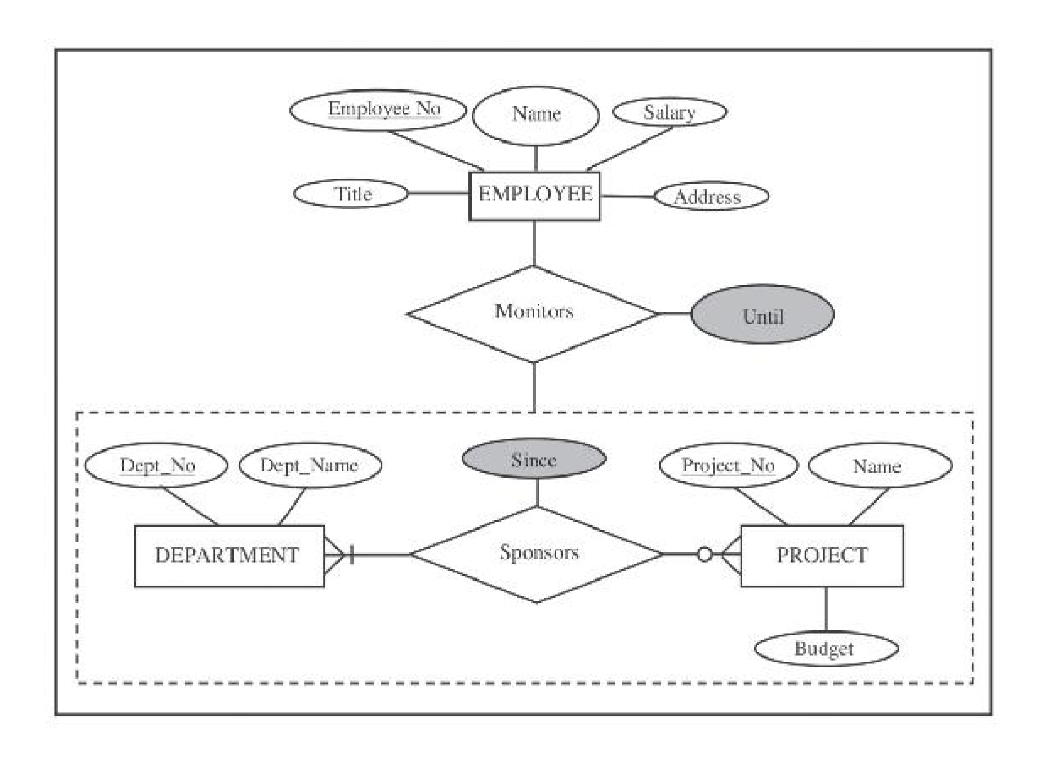
عُرِّفت فئة العلاقة بأنها ارتباط بين نوعين أو أكثر من فئات الكينونات. غير أنه في بعض الأحيان نحتاج إلى نمذجة فئة علاقة بين فئة كينونة ما، من جانب، ومجموعة من الكينونات والعلاقات مجتمعة مع بعضها، من جانب آخر. فعلى سبيل المثال: لنفترض وجود فئة كينونة بمسمَّى «مشروع» (PROJECT) وفئة كينونة بمُسمَّى «قسم» (DEPARTMENT)، وأن كلَّ مشروع «مدعوم مالياً» (Sponsored)، من خلال قسم واحد أو أكثر، وأن القسم الواحد يمكن ألا يدعم مالياً أيَّ مشروع أو أنه يدعم أكثر من مشروع. وعندما يبدأ قسم بدعم مشروعٍ ما، يُوجَد

هناك تاريخ لبداية الدعم. يمكن نمذجة هذا الوضع من خلال علاقة «يدعم مالياً»، كما هو موضح في الشكل رقم (3-19).



شكل رقم (3–19): علاقة «الدعم المالي» التي تربط بين فئة كينونة الأقسام وفئة كينونة المشاريع.

لنفترض الآن أنه كلما قام قسمٌ بدعم مشروعٍ ما؛ فإنه يقوم أيضاً بتكليف موظف (أو مجموعة موظفين) لمتابعة سير المشروع. من المنطقي في هذه الحالة أن علاقة «المتابعة» (Monitors) تربط بين فئة كينونة الموظفين، من جانب، وعلاقة «الدعم المالي» (Sponsors)، وليس فئة كينونة المشروع أو فئة كينونة القسم. لهذا السبب يمكن استخدام «التجميع» (Aggregation) الذي يمكننا من تمثيل مثل هذا الوضع (Ramakrishnan and Gehrke, 2003). ويمكِّن التجميع من توضيح أن فئة علاقة (عوضاً عن كينونة) ترتبط بفئة كينونة من خلال فئة علاقة أخرى. وتمثل فئة العلاقة المجمعة؛ من خلال وضعها داخل مستطيل منقط الأضلاع. ويُوضِّح الشكل رقم (3-20) علاقة «الدعم المادي» والكينونات التي تربط بينها بعد تجميعها (من خلال وضعها داخل مستطيل منقط الأضلاع)؛ للدلالة على أنها تدخل مجتمعةً في علاقة أخرى هي علاقة «المتابعة».

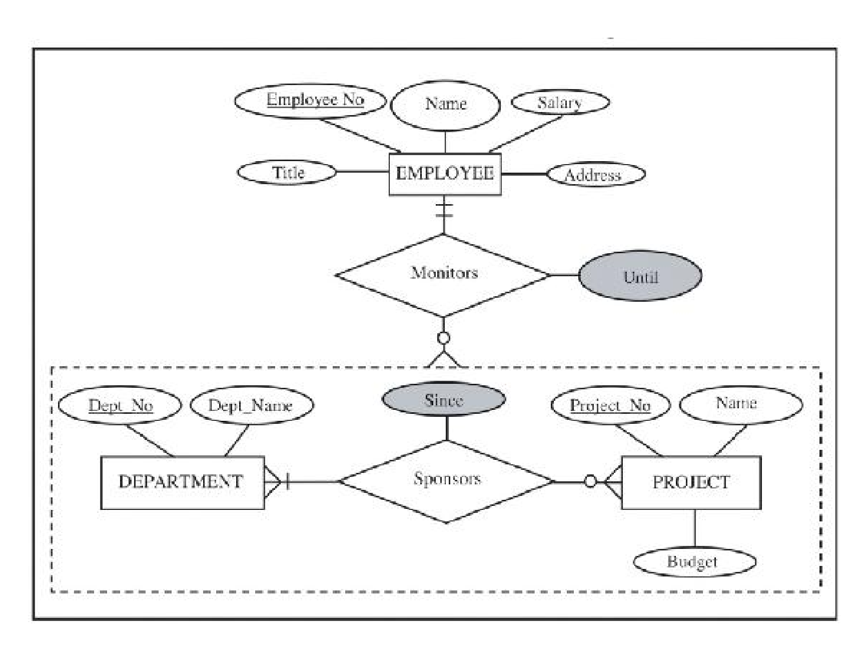


شكل رقم (3–20): علاقة «متابعة سَيْر المشروع» عند استخدام مفهوم التجميع.

يمكِّننا - إذن - التجميع من معاملة علاقة ما، مثل «الدعم المالي» (Sponsors)، وكأنها فئة كينونة عند تعريف علاقة أخرى، مثل «المتابعة» (Monitors) أعلاه. ويمكن استخدام التجميع، بـشكلٍ عام، عند محاولة نمذجة علاقة من ضمن أطرافها علاقة (أو علاقات) أخرى. ولكن هل بالإمكان عدم استخدام التجميع عندما يكون أحد أطراف علاقة ما علاقة أخرى والاستعاضة عنه باستخدام علاقة ثلاثية أو علاقة ثنائية؟ والإجابة عن هذا السؤال هو عدم إمكانية ذلك في مثل الحالة التي افترضناها أعلاه؛ إذ إنه لا يمكن استخدام علاقة ثلاثية تربط بين الكينونات الثلاث؛ وذلك لوجود علاقتين مستقلتين هما «الدعم المالي» و«المتابعة». ليس هذا فحسب، ولكن كل علاقة من العلاقتين قد يكون لها خصائصها المتعلقة بها مثل تاريخ «بداية» الدعم المالي (SinCe) للمشروع التي ترتبط بعلاقة «الدعم المالي» وخاصية «نهاية» فترة المتابعة (Until) للموظف التي ترتبط بعلاقة «المتابعة».

كما أنه لا يمكن الاستعاضة عن التجميع من خلال علاقة ثنائية، وهي علاقة «المتابعة» التي تربط بين كينونة المشروع وكينونة الموظف، على سبيل المثال. ويُعزَى السبب وراء ذلك إلى كون علاقة المتابعة لا ترتبط بالمشروع؛ ولكنها ترتبط بعلاقة «الدعم المالي».

ويمكِن إيضاح قيود التعدُّدية على علاقة المتابعة كما لو كان التجميع فئة كينونة. فعلى سبيل المثال، لنفترض «أن الموظف الواحد قد لا يتابع أيَّ دعمٍ ماليٍّ أو أنه يقوم بمتابعة أكثر من دعمٍ ماليّ واحد، وأن كل دعم مالي تجب متابعته من قِبل موظف واحد فقط». في هذه الحالة، يمكن تمثيل قيود التعددية كما هو مُوضَّح في الشكل رقم (3-21).



شكل رقم (3–21): إيضاح التعدُّدية عند استخدام مفهوم التجميع في نمذجة العلاقات.

حالة دراسية

قاعدة بيانات شركة عقارية

(الحل متوفر في الملحق رقم 2)

تعتزمُ إحدى الشركات العقارية الكبرى مكننة عملها الإداري؛ من خلال تطوير نظم تطبيقية تعتمدُ في بنائها على نظم قواعد البيانات. وكخطوةٍ أولى تعتزم الشركة نمذجة بعض قواعد العمل المعمول بها في الشركة والتي تحكم طبيعة عملها باستخدام النموذج المفاهيمي «كينونة - علاقة». باستخدام قواعد العمل التالية، المطلوب هو تصميم قاعدة البيانات باستخدام نموذج «كينونة - علاقة».

1- يُوجَد للشركة مجموعة من المكاتب (OffiCes) في مناطق ومدن مختلفة، ولكل مكتب رمز (OffiCe_ID) يميزه بشكلٍ منفرد عن بقية المكاتب التابعة للشركة وعنوان (Address)، ورقم هاتف (Telephone_No).

2- يعمل في الشركة مجموعة من الموظفين، ولكلِّ موظف رمز وظيفي (Employee_ID) يميزه عن بقية الموظفين العاملين في الشركة، واسم يتكون من (الاسم الأول، واسم الأب، واسم العائلة)، وتاريخ ميلاد (DOB)، وعنوان سكني، ورقم هاتف، وشهادة علمية أو أكثر (ACademiC_Degree) تتكون من الدرجة العلمية (Degree)، وتاريخ الحصول عليها (Date)، ومكان الحصول عليها (Issuing_Institution).

3- يعمل كلُّ موظف في مكتب واحد فقط في أية فترة زمنية. وعندما يبدأ الموظف في العمل في مكتب ما؛ فإن هنالك تاريخاً لبدئه للعمل في المكتب (Starting_Date). كما أنه قد يعمل الموظف في أكثر من مكتب؛ ولكن في فترات زمنية مختلفة وغير متداخلة مع بعضها. ويعمل في المكتب الواحد موظف واحد أو أكثر.

4- تشرف الشركة على مجموعة من العقارات، ولكل عقار رمزٌ (Estate_ID) يميِّزه بشكلٍ منفردٍ عن بقية العقارات التي تشرف عليها الشركة، وموقع (Address). والعقار الواحد

الذي تشرف عليه الشركة يجب أن يكون إما للإيجار (Rental_Estate) أو للبيع (Sale_Estate). وعندما يكون العقار معروضاً للبيع؛ فإنه ليس من الممكن أن يكون للإيجار، كما أن العقار المعروض للإيجار لا يمكن أن يكون قابلاً للبيع. ولكل عقار معروض للإيجار تاريخ (Date) يوضِّح اليوم الذي من الممكن أن يبدأ به إيجار العقار؛ بحيث أنه لا يمكن تأجير العقار قبله، وسعراً لإيجاره السنوي (Yearly_Rent)، ومبلغاً للتأمين عليه (InsuranCe_Amount). أما العقار المعروض للبيع فله سعر (PriCe) ونسبة لعمولة الشركة من مبلغ البيع، وهي 5.2% من سعر بيع العقار.

5- يُشرف كلُّ مكتب من مكاتب الشركة على واحد أو أكثر من العقارات؛ (سواءً كانت معروضة للبيع أو للإيجار)، ويجب أن يتمَّ الإشراف على العقار من قِبل مكتب واحد فقط من مكاتب الشركة.

6- يتعامل مع الشركة مجموعة من العملاء (Customers). ولكلِّ عميل رمز يميزه عن بقية العملاء (Customer_ID)، واسم، ورقم تليفون، وعنوان بريدي. والعميل الواحد إما أن يكون مالكاً لعقار (يرغب في بيعه أو في إيجاره) (Owner) أو مشترياً لعقار (Buying_Customer) أو مستأجراً لعقار (Renting_Customer).

7- يجب أن يملك (Owns) كلَّ عقارٍ واحدٌ أو أكثر من المالكين، والمالك الواحد قد يكون له واحد أو أكثر من العقارات (المعروضة للبيع أو للإيجار). ولكل مالك لعقار تاريخ (Date) يبيّن ملكيته له ورقم الصك (Estate_Ownership_ID) الذي تملَّك من خلاله العقار. أما العقارُ المعروض للبيع؛ فله صفر أو أكثر من المشترين، ولكل مشترٍ صفر أو أكثر من العقارات المشتراة من الشركة.

8- عندما يتمُّ بيع عقار لأحد المشترين؛ فإن لعملية البيع تاريخاً (Date) وسعراً للبيع (Sale_PriCe) قد يكون مختلفاً عن السعر الذي تمَّ تحديده بشكلٍ مبدئي كقيمة للعقار. أمَّا العقار المعروض للإيجار؛ فيتم إيجاره وفق عقد (Lease_Agreement) يتحَدَّد فيه المستأجر والعقار، كما يتضمَّن العقد تاريخ بدء سريان عقد الإيجار (Lease_Date) الذي قد يكون مختلفاً عن التاريخ الذي عرض فيه العقار للإيجار.

الفصل الرابع

النموذج العلاقي ولغاته الرسمية

يُركِّز هذا الفصل من الكتاب على شرح المفاهيم الأساسية للنموذج العلاقي الذي يُعَدُّ أنجح النماذج التمثيلية للبيانات، وأكثرها استخداماً في نظم قواعد البيانات المتوافرة على المستوى التجاري. ومن أبرز أسباب نجاح وانتشار استخدام هذا النموذج سهولته في تمثيل البيانات؛ إضافةً إلى استناده إلى أسس رياضية صلبة تمكِّنه من التعامل مع البيانات وحساب نتائجها. لذلك؛ فإن هذا الفصل يستعرض أيضاً لغتين من لغات النموذج العلاقي الرسمية، وهما: «الجبر العلاقي» (Relational Algebra) - التي تُعَدُّ إحدى لغات النموذج العلاقي الإجرائية (ProCedural Language)؛ و«الحساب العلاقي» (Relational CalCulus) - التي تُعَدُّ إحدى لغات النموذج العلاقي غير الإجرائية (NonproCedural Language). كما أن من أسباب نجاح هذا النموذج وجود لغة قياسية (Standard Language) للتعامل مع هذا النموذج، وهي لغة الاستفسار البنائية (StruCtured Query Language (SQL)) التي يمثل شرح مكوناتها محتوى كلٍّ من الفصلين السابع والثامن.

4-1 نموذج البيانات العلاقي (**Relational Data Model**):

ظهر النموذج العلاقي عام 1970م على يد «إدغار كود» الذي كان يعمل في شركة أي بي إم (IBM)؛ وذلك من خلال ورقته العلمية الشهيرة التي قام بنشرها عام 1970م (Codd, 1970). ولقد لاقى هذا النموذج اهتماماً كبيراً منذ بداية ظهوره؛ وذلك لسهولته في تمثيل البيانات، ولوجود أسس رياضية يعتمد عليها. وتم البدء في بناء نظامين تجريبيين بحثيين؛ للتحقق من جدوى النموذج العلاقي: الأول منهما كان في أحد مراكز شركة أي بي إم للبحوث (IBM San Jose ResearCh

Laboratory) الذي تمَّ فيه تطوير نظامٍ عُرِف «بنظام آر» (System R) في أواخر السبعينيات الميلادية. أما الثاني؛ فكان نظاماً ذا طبيعة أكاديمية، وتمَّ تطويره في جامعة بيركلي الأمريكية، وعُرِف بنظام «إنجرس» (Ingres). ومع بداية الثمانينيات الميلادية؛ بدأت تظهر نظم إدارة قواعد بيانات عديدة تعتمد في بنائها على النموذج العلاقي. أما اليوم؛ فإن نظم قواعد البيانات العلاقية هي السائدة، وهي تُراوح في استخداماتها بين تلك التي يمكن تركيبها واستخدامها على «الحاسبات الشخصية» (Personal Computers) وصولاً إلى تلك التي يتمُّ تركيبها واستخدامها على «الحاسبات الكبيرة» (أو المركزية) (Mainframes).

4–1–1 المفاهيم الأساسية في النموذج العلاقي:

تُمثَّل البيانات في نموذج البيانات العلاقي على هيئة مجموعةٍ من الجداول تُسمَّى علاقات (Relations). وكل جدول في النموذج العلاقي يحتوي على مجموعةٍ من الصفوف ومجموعةٍ من الأعمدة؛ بحيث إن كلَّ صفٍّ من صفوف الجدول يمثل بيانات حالة من الحالات التي لها مفهومها في المنظمة، وكل عمودٍ يمثل إحدى خصائص هذه الحالة. ويتكون النموذج العلاقي من ثلاثة مكونات رئيسية (Fleming and von Halle, 1989)، وهي:

1- هياكل البيانات (Data StruCtures): يتم تنظيم البيانات ضِمن جداول (تُسمَّى رسمياً علاقات في النموذج العلاقي) تتكون من صفوف (تُسمَّى (Tuples) باللغة الإنجليزية) وأعمدة (تُسمَّى (Attributes) باللغة الإنجليزية).

2- عمليات للتعامل مع البيانات (Data Manipulation Operations): يتوفَّر للنموذج العلاقي لغات تمكِّن من التعامل مع بيانات النموذج. وهذه اللغات مبنيةٌ على أسس رياضية صلبة (مثل: الجبر العلاقي، والحساب العلاقي).

3- تكامل البيانات (Data Integrity): يتوفر في النموذج العلاقي تسهيلات تمكِّن من توصيف (بعض) قواعد العمل التي تُمكِّن من المحافظة على تكامل البيانات.

وفيما يلي شرحٌ لهياكل البيانات وتكامل البيانات في النموذج العلاقي. أمَّا لغات التعامل (الرسمية) مع النموذج العلاقي؛ فسيتم شرحها في الجزأين التاليين من هذا الفصل.

4-1-1-1 هيكل البيانات العلاقي:

تُمثِّل العلاقة الهيكل الأساسي للبيانات في النموذج العلاقي الذي يستمدُّ مُسمَّاه من مسمى الهيكل الرئيسي لبياناته. والعلاقة هي جدول بسيط له مُسمَّى. وتتكون العلاقة (أو الجدول) من عددٍ مُحدَّد من الحقول (أو الأعمدة) لها مسميات، وعددٍ غير مُحدَّد من الصفوف دون مسميات لها. ويُمثِّل عدد الحقول «درجة الجدول» (Degree (or Arity)). وكل صف في الجدول يمثل سجلاً (ReCord) لإحدى الحالات؛ بحيث يحتوي على قيم للبيانات في كل عامود من الأعمدة التي يحتوي عليها الجدول. ويعني هذا أن كل عامود يمثل إحدى خصائص الحالات المدوَّنة في الجدول ويحمل اسم الخاصية. ويوضِّح الشكل رقم (4-1) جدول «أعضاء هيئة التدريس» (FACULTY_T) وفق النموذج العلاقي، الذي يكافئ فئة كينونة «أعضاء هيئة التدريس» في النموذج «كينونة - علاقة».

FACULTY_T

Faculty_ID	FName	LName	Phone_NO	Salary	DOB
200	Khalid	Aloufi	454-2341	35000	22/05/1963
220	Fahad	Alhamid	456-7733	25900	07/10/1970
310	Saleh	Aleesa	454-8932	30000	13/09/1966
320	Mohammed	Alhamad	454-5412	44000	13/05/1965
330	Ghanim	Alghanim	456-2234	44500	12/08/1969
340	Ibraheem	Alsaleh	454-1234	25000	20/01/1970
400	Ahmad	Alotaibi	454-4563	33900	17/05/1971
420	Saleh	Alghamdi	454-2233	44600	13/02/1969
500	Yahya	Khorshid	456-2221	36700	12/03/1965
540	Salem	Alhamad	456-3304	40000	11/09/1972
560	Salman	Albassam	454-7865	33800	13/09/1968
600	Turki	Alturki	456-7891	27800	23/07/1975
640	Fahad	Alzaid	456-3322	44300	12/05/1971
660	Saud	Alkhalifa	454-9856	44900	13/08/1972
710	Mahmood	Alsalem	456-3323	31900	19/02/1973
730	Mishal	Almazid	454-2343	29800	17/09/1975
770	Sultan	Aljasir	456-3212	43300	13/05/1970
800	Ali	Albader	456-7812	45300	22/06/1966
810	Saad	Alzhrani	454-5578	44200	17/10/1967
850	Ahmad	Alsabti	456-0120	33900	15/04/1973

شكل رقم (4-1): جدول «أعضاء هيئة التدريس» (FACULTY_T) وفق النموذج العلاقي.

ويتكون الجدول في الشكل السابق من ستة حقول تمثل الخصائص التالية: رمز عضو هيئة التدريس، والاسم الأول لعضو هيئة التدريس، واسم العائلة، ورقم الهاتف، والمرتب، وتاريخ الميلاد.

وعليه؛ فإن درجة الجدول هي ستة (6). أما كلُّ صف من صفوف الجدول؛ فيمثل حالةً واحدةً من حالات أعضاء هيئة التدريس. وقد تمَّ إدراج عددٍ من حالات أعضاء هيئة التدريس في الجدول لإيضاح هيكل الجدول؛ إذ إن الحالات نفسها لا تُعَدُّ من ضِمن هيكل الجدول. كما أن هذه الحالات تتغير بشكلٍ مستمرٍ من خلال عمليات «التعديل» (ModifiCation)، وهي: «الإضافة» (Insert)، و«الحذف» (Delete)، و«التحديث» (Update). ويعني هذا أن السجلات الموجودة في الجدول، في أية لحظة ما، تمثل حالة من حالات الجدول في تلك اللحظة، والتي قد تتغير في لحظة أخرى. ومن الممكن أن تكون حالة الجدول فارغة بمعنى عدم وجود أية سجلات (أو صفوف) فيه.

ويتمُّ في بعض الأحيان تمثيل هيكل الجدول بشكلٍ مختصرٍ من خلال كتابة اسم الجدول متبوعاً بأسماء حقول الجدول التي تتمُّ كتابتها بـين قوسين. فعلى سبيل المثال: يتم تمثيل جدول «أعضاء هيئة التدريس» أعلاه وفق الطريقة المختصرة هذه، كما يلي:

FACULTY_T (FaCulty_ID, FName, LName, Phone_NO, Salary, DOB)

4-1-1-2 المفاتيح في النموذج العلاقي:

يُعرَّف الجدول (أو العلاقة) في النموذج العلاقي على أنه مجموعة من السجلات (Tuples). ولأن المجموعات في تعريفها الرياضي (وفقاً لنظرية المجموعات الحسابية) لا تحتوي على قيم متكررة؛ يجب في السجلات التي تخزن في أيِّ جدول علاقي ألا تتكرر في نفس الجدول؛ حتى ينطبق عليها تعريف المجموعة. ويعني هذا أنه لا يمكن أن يكون لسجلين في جدول علاقي نفس التوليفات في قيم خصائصهما. ويُوجَد عادة مجموعة من الخصائص في أيِّ جدول علاقي لا يمكن أن تتكرَّر بـين سجلات الجدول. وتُسمَّى هذه الخصائص التي تميز بين السجلات المختلفة في الجدول «المفتاح الخارق» (Superkey (SK)). ويعني هذا أنه لأيِّ سجلين (Tuples)، وليكونا 1t و 2t في جدول علاقي ما، وليكن اسمه "R"، وعلى افتراض أن المفتاح الخارق للجدول هو "SK"؛ فإنه لا بد أن يتحقق الشرط التالي:

$$t1\ [SK] \neq t2\ [SK]$$

ويعني الشرط أعلاه أنه لا يمكن أن يُوجَد في جدول علاقي ما، له مفتاح خارق اسمه "SK" يتكون من مجموعةٍ من الخصائص، أن يتكرَّر في سجلين هما 1t و 2t. كما يعني هذا أن أيَّ مجموعة من الخصائص في الجدول تحقق الشرط أعلاه؛ تُعَدُّ مفتاحاً خارقاً للجدول. وذلك يعني أنه بالإمكان أن يتوفر في الجدول أكثر من مفتاح خارق. ويضمن مبدأ المفاتيح الخارقة تفرُّد السجلات في الجداول العلاقية؛ بحيث إنه لا يمكن أن يُوجَد سجلان في جدول ما، وفي أية حالة من حالات الجدول، أن يكون لهما نفس قيمة المفتاح الخارق. ولكل جدول علاقي مفتاح خارق واحد على الأقل. وبذلك يكون المفتاح الخارق الافتراضي لأيَّة جدول علاقي مكوناً من كافة حقول الجدول.

ولعدم نصِّ تعريف المفتاح الخارق على أن الحقول المكونة للمفتاح الخارق يجب أن تكون أقل ما يمكن من حقول تمكِّن من التعرُّف على سجلات الجدول بشكلٍ منفردٍ؛ فإنه قد يكون من ضِمن الحقول المكوِّنة للمفتاح الخارق حقولٌ لا تؤثر في عملية التعرُّف على سجلات الجدول بشكلٍ منفرد. لذا؛ فإن هذا يقودنا إلى تعريف مبدأ «المفتاح» الذي يُعَدُّ أكثر فائدةً من مبدأ المفتاح الخارق. و«المفتاح» في أية علاقة هو «مفتاح خارق»؛ ولكننا لا نستطيع حذف أيِّ حقل من الحقول المكوِّنة له، وفي الوقت نفسه، الاستمرار في التعرُّف على سجلات الجدول بشكلٍ منفرد. لذا؛ فإن أيَّ مفتاحٍ لجدول علاقي ما يجب أن يحقق الشرطين التاليين:

1- لا يمكن أن تتكرَّر قيم (كافة) الحقول المكوِّنة للمفتاح، في أية حالة من حالات الجدول، لسجلين مختلفين في الجدول.

2- لا يمكن حذف أيِّ حقلٍ من الحقول المكوِّنة للمفتاح، وفي الوقت نفسه، الاستمرار في التعرُّف على سجلات الجدول بشكل منفرد.

ينطبق الشرط الأول على كلٍّ من «المفتاح الخارق» و «المفتاح». أمَّا الشرط الثاني؛ فينطبق على «المفتاح» فقط. كما يعني الشرطان مع بعضهما أن «المفتاح» هو «مفتاح خارق»، يميز بين سجلات الجدول بشكلٍ منفرد؛ ولكنه يتكون من أقلِّ عددٍ ممكنٍ من الحقول التي تميِّز بين سجلات الجدول بشكلٍ منفردٍ، وفي أية حالة من حالات الجدول. فعلى سبيل المثال: يمثل حقل «رمز عضو هيئة التدريس» (FaCulty_ID) مفتاحاً لجدول «أعضاء هيئة التدريس» أعلاه؛ لكونه يميِّز بين سجلات الجدول، المتعلقة بأعضاء هيئة التدريس، بشكلٍ منفردٍ ليس في حالة الجدول أعلاه فحسب؛ ولكن في أية حالة ممكنة من الحالات التي قد يكون عليها الجدول. كما يُعَدُّ رمز عضو هيئة

التدريس مع أيَّة حقول أخرى في الجدول مفتاحاً خارقاً. فعلى سبيل المثال: يُعَدُّ حقل رمز عضو هيئة التدريس والاسم الأول مفتاحاً خارقاً للجدول، ورمز عضو هيئة التدريس واسم العائلة مفتاحاً خارقاً آخر للجدول، ورمز عضو هيئة التدريس، والاسم الأول، والمرتب؛ مفتاحاً خارقاً ثالثاً للجدول... وهكذا. وهذا يعني أنَّ كل «مفتاح» هو «مفتاح خارق»؛ ولكن ليس كل «مفتاح خارق» يكون «مفتاحاً». فعلى سبيل المثال: عند حذف حقل الاسم الأول من المفتاح الخارق المكوَّن من حقل رمز عضو هيئة التدريس والاسم الأول؛ ما زلنا نستطيع التعرُّف على كلِّ سجل من سجلات الجدول بشكلٍ منفرد. لذا؛ فإن الشرط الثاني أعلاه لا ينطبق على هذا المفتاح؛ وبالتالي فإنه «مفتاح خارق» ولكنه ليس «مفتاحاً» للجدول. تجدر الإشارة هنا أنه ليس من الضروري أن يتكون أيُّ مفتاحٍ (سواء كان خارقاً أو مفتاحاً فقط) من حقلٍ واحدٍ فقط؛ إذ إنه في الكثير من الأحيان يكون مفتاح الجدول «مفتاحاً مركباً» يتكوَّن من أكثر من حقل من حقول الجدول.

ويُعَدُّ وجود مفتاح في أيِّ جدول علاقي خاصية من خصائص هياكل الجداول في النموذج العلاقي بدونها لا يُعَدُّ الجدول علاقياً. ويمكن تحديدُ المفتاح لأيِّ جدول من خلال خصائص الحقول المكونة له؛ بحيث يجب أن تكون من ضمن خصائص هذه الحقول عدم تغيُّر قِيمها بتغيُّر الزمن بالإضافة إلى كونها تميِّز بين سجلات الجدول بشكلٍ منفرد. فعلى سبيل المثال: لا يمكِن أن يكون رقم الهاتف مفتاحاً لجدول أعضاء هيئة التدريس؛ لأنه قد يتغيَّر بتغيُّر الزمن. كذلك هو الحالُ بالنسبة لحقل الاسم الأول واسم العائلة؛ فهذان الحقلان مُدمجان مع بعضهما لا يصلحان أن يصبحا مفتاحاً للجدول؛ لأنهما قد يتكرَّران في أكثر من سجلٍ من سجلات أعضاء هيئة التدريس. أما رمز عضو هيئة التدريس؛ فله قيمة لا تتكرر بين أعضاء هيئة التدريس، وفي الوقت نفسه، لا تتغير بتغيُّر (أو مرور) الزمن.

وبشكلٍ عام؛ فإنه قد يُوجَد في هيكل أيِّ جدولٍ علاقي أكثر من مفتاح. في هذه الحالة يُعَدُّ كلُّ مفتاح من هذه المفاتيح «مفتاحاً مرشحاً». فعلى سبيل المثال: وفي حالة تمَّ إدراج رقم السجل المدني ليصبح حقلاً ضِمن جدول أعضاء هيئة التدريس أعلاه؛ فإن كلاً من رمز عضو هيئة التدريس ورقم السجل المدني لعضو هيئة التدريس يُعدَّان مفاتيح مرشحة في جدول أعضاء هيئة التدريس، تنطبق على كلٍّ منهما شروط المفتاح. وإذا وُجِدَ أكثرُ من مفتاح لجدولٍ ما؛ فإنه يتمُّ اختيار إحدى هذه المفاتيح المرشَّحة ليصبح «المفتاح الرئيسي» للجدول. ويعني هذا أن «المفتاح الرئيسي»؛ هو أحد المفاتيح المرشحة الذي تمَّ اختياره ليميِّز بين السجلات المختلفة في الجدول. ومن المتعارف

عليه عند اختيار المفتاح الرئيسي من ضِمن مجموعةٍ من المفاتيح المرشَّحة؛ هو اختيارُ المفتاح المرشَّح ذي العدد الأقل من الحقول. وعند تمثيل المفتاح الرئيسي في هيكل الجدول؛ جرَت العادة على أن يُوضَع خط تحت الحقل المكون (أو الحقول المكونة) للمفتاح الرئيسي؛ وذلك للتمييز بين حقول المفتاح الرئيسي وبقية حقول الجدول. أما بالنسبة لبقية المفاتيح المرشحة؛ فإنها تُسمَّى أيضاً «مفاتيح ثانوية» (SeCondary Keys) قد يتمُّ استخدام بعضٍ منها في إنشاء فهارس ثانوية للجدول في أثناء عملية بناء قاعدة البيانات (في مرحلة التصميم المادي)؛ وذلك بهدف التسريع في عملية البحث في سجلات الجدول واسترجاعها.

4-1-1-3 خصائص العلاقات (أو الجداول) في النموذج العلاقي:

تمَّ تعريف العلاقة أعلاه على أنها جدول بسيط؛ ولكنه من الضروري معرفة أنه ليس كل جدول بسيط يمثل علاقةً وفقاً للنموذج العلاقي. ولهذا السبب لم يتم استخدام كلمة «جدول» في النموذج العلاقي بشكله الرسمي، وإنما تمَّ استخدام كلمة «علاقة». فالعلاقات لها عددٌ من الخصائص التي تميّزها عن الجداول غير العلاقية (أو التقليدية) التي نتعامل معها في حياتنا اليومية. وفيما يلي استعراض لخصائص العلاقات (أو الجداول العلاقية):

1- كلُّ جدول علاقي يجب أن يكون له اسم.

2- كلُّ خاصية (أو عمود) لها اسمٌ فريدٌ يميّزها عن بقية الخصائص (أو الأعمدة) المعرفة في نفس الجدول.

3- كلُّ صفٍّ في الجدول يُعَدُّ فريداً لا يمكن أن يتكرَّر ضِمن الجدول.

4- كلُّ تقاطع بين صفٍّ وعمودٍ (خلية من خلايا الجدول) يحتوي على قيمة واحدة فقط، بمعنى أنه لا يمكن إدخالُ أكثر من قيمة داخل نفس الخلية.

5- لا يُعَدُّ ترتيب الحقول مهماً؛ إذ إنه يمكن تغيير ترتيبها دون الإخلال بمعنى أو طريقة استخدام الجدول.

6- لا يُعَدُّ ترتيب السجلات (أو صفوف) الجدول مهماً؛ إذ إنه يمكن تغيير ترتيبها أو تخزينها، كما هو الحال بالنسبة للحقول، دون الإخلال بمعنى أو طريقة استخدام الجدول.

وتعني الخصائص السابقة للجداول العلاقية أن كلَّ جدول يحتوي على بيانات تمثل مجموعةً من الحقائق. فعلى سبيل المثال: يحتوي جدول «أعضاء هيئة التدريس» على بيانات تمثل حقائق عن أعضاء هيئة التدريس. فكلُّ صفٍ في الجدول يمثل حقيقةً عن بيانات أحد أعضاء التدريس. ويعني هذا أن تغيير ترتيب الصف (أو السجل) الممثل لأحد أعضاء هيئة التدريس ضِمن صفوف الجدول لن يغيّر من البيانات الخاصة بعضو هيئة التدريس. كذلك هو الحالُ لو تم تغيير ترتيب الأعمدة (أو الحقول) المكونة للجدول؛ فإن مثل هذا التغيير لن يغير من البيانات التي تمثل حقائق عن أعضاء هيئة التدريس. أمَّا بالنسبة لمسميات الأعمدة (أو الحقول) في جداول النموذج العلاقي فهي مهمة؛ لأنها تدل على معنى البيانات (أو الحقائق) التي يحتويها كل حقلٍ وكذلك للتفريق بين معنى حقل ما وبقية الحقول في الجدول. ووجود مُسمَّى لكل جدول يمكِّننا من التعامل مع الحقائق المختلفة الموجودة في قاعدة البيانات من خلال لغات تداول البيانات في النموذج العلاقي التي سنتطرق لها لاحقاً. وبدون هذه المسميات للجداول لا يمكن التعرُّف على الجدول الذي سيتمُّ التعامل معه.

وتجدرُ الملاحظة أن الجداول العلاقية قد تمثل حقائق عن الكينونات أو العلاقات بين الكينونات حسب تعريفها في نموذج «كينونة - علاقة». فعلى سبيل المثال: يتم تمثيل فئة كينونة «أعضاء هيئة التدريس» وكينونة «المواد الدراسية» كجدولين منفصلين في النموذج العلاقي لتمثيل الحقائق التي تتعلق بكلٍ من أعضاء هيئة التدريس والمواد الدراسية، على التوالي. أما العلاقة التي تربط بين الكينونتين وهي علاقة «مؤهل لتدريس»؛ فإنه يتمُّ تمثيلها أيضاً لتصبح جدولاً في النموذج العلاقي تمثل الحقائق المتعلقة بالمواد الدراسية المؤهل لتدريسها كلُّ عضو هيئة تدريس. ويعني هذا أن كلاً من الكينونات والعلاقات (ذات تعدُّدية متعدد - متعدد) تُمثَّل في النموذج العلاقي بشكلٍ متماثلٍ على هيئة جداول.

4-1-1-4 قيود التكامل (Integrity Constraints) في النموذج العلاقي:

تحتوي أيَّة قاعدة بيانات علاقية على العديد من الجداول. وترتبط السجلات الموجودة في الجداول المختلفة المكوِّنة لقاعدة البيانات بأشكالٍ متنوعة؛ حتى تستطيع أن تمثل الواقع الذي نحاول تمثيله. وتُمثِّل كل «حالة» من حالات قاعدة البيانات «حالات» كافة جداول قاعدة البيانات في لحظةٍ معينةٍ من الزمن. ويُوجَد عادةً الكثيرُ من القيود التي يجب أن تنطبق على قاعدة البيانات في

أية حالة من حالاتها. وتُمثل هذه القيود قواعد العمل المعمول بها في المنظمة التي نحاول تمثيل بـياناتها وفق النموذج العلاقي. فعلى سبيل المثال: عندما نقول: إن كلَّ عضو هيئة تدريس لا بد أن يتبع (أو يعمل) في قسم دراسي واحد فقط؛ فإن مثل قاعدة العمل هذه تمثل قَيْداً على كافة حالات قاعدة البـيانات. ويعني هذا أنه في أيَّة حالة من حالات قاعدة البيانات لا بد أن يرتبط كلُّ عضو هيئة تدريس بقسم دراسي يمثل مقرَّ عمله؛ حتى تكون قاعدة البـيانات صحيحةً ومتكاملةً. وفيما يلي شرحٌ لأهمِّ أنواع القيود التي يُمكِن تمثيلها في النموذج العلاقي. وهذه القيود هي: «قيود المجال» (Domain Constraints)، و«قيود تكامل الجدول» (Entity Integrity Constraints)، و«قيود القيم الغائبة» (NULL Constraints)، و«قيود السلامة المرجعية» (Referential Integrity Constraints).

4-1-1-4-1 قيود المجال (**Domain Constraints**):

عندما يتمُّ تعريف حقل ضِمن جدول علاقي؛ فإنه لا بد أن يتم تحديد نوعية البـيانات التي سيتخذ الحقل قيماً منها. وهذه الحالة شبيهةٌ بتعريف «المتغيرات» (Variables) في لغات البرمجة؛ إذ إن أيَّ متغير لا بد أن يتم تحديد نوعية لبياناته (وذلك في الغالبية العظمى من لغات البـرمجة). أما «المجال» (Domain)؛ فيضيف إلى ذلك تحديد مدى (أو مجموعة مُحدَّدة) من القِيم التي من الممكن أن يتخذ الحقل قيماً منها. لذا؛ فإن أيَّ مجال في النموذج العلاقي لا بد أن يرتبط بنوعية بيانات محددة وبـمدى (أو مجموعة) قيم. فعلى سبيل المثال، يمكن تحديد نوعية بيانات حقل «راتب عضو هيئة التدريس» في جدول أعضاء هيئة التدريس على أنه من نوع «الأعداد الصحيحة» (Integers). غير أنه في غالبية المنظمات يُوجَد حَدٌّ أعلى وحَدٌّ أدنى للرواتب التي يتقاضاها الموظفون. في هذه الحالة يمكن تحديد مدى القيم المسموح بها لأن تكون رواتب لأعضاء هيئة التدريس؛ وليكن بين 000.5 و 000.30. وبهذه الطريقة؛ فإن النموذج العلاقي يُمكِّن من وَضْع قيود على مدى القيم التي من الممكن أن يأخذها أيُّ حقل؛ مما يسهم في صحة البيانات وتكاملها؛ لأن محاولة إدخال أية قيمة تخرج عن المجال، أيْ: نوعية البيانات ومدى القيم المسموح بها، الذي تم تحديده لحقلٍ ما؛ لن تكون مقبولةً في النموذج العلاقي.

ويتكوَّن تعريف أيِّ مجال لحقلٍ ما، في العادة، من: اسم المجال (Domain Name)، ومعناه (Meaning)، ونوعية بياناته (Data Type)، وطوله (Size)، ومدى (أو مجموعة) القيم المسموح

بـها. ويُسهِم استخدام اسم لأيِّ مجال في تسهيل عملية تفسير القيم التي من الممكن أن يأخذها أيُّ حقل يرتـبط بـه. ويوضح الجدول رقم (4-1) تعاريف لمجال بعض الحقول.

جدول رقم (4-1): أمثلة لتعاريف مجال بعض الحقول.

الحقل (أو العمود)	"اسم المجال" الذي يرتبط به الحقل	معنى المجال	مدى القيم
Department_ID	Department_IDs	مجموعة القيم التي من الممكن أن يأخذها حقل "رمز القسم الدراسي"	نصي بطول ستة أحرف: Char(6)
Gender	Gender_Type	جنس عضو هيئة التدريس: ذكراً أو أنثى	نصي بطول حرفٍ واحد من القيم "M" للذكر و "F" للأنثى: Char(1) In (‘M’, ‘F’)
Salary	Salary_Range	راتب عضو هيئة التدريس	عدد صحيح بين 000.5 و 000,30: Integer Between 5,000 and 30,000
Course_ID	Course_IDs	رمز المادة الدراسية	نصي بطول سبعة أحرف: Char(7)

4–1–1–4–2 قيود تكامل الجدول (أو العلاقة) (Entity Integrity Constraints):

إنَّ قيد تكامل الجدول (أو العلاقة) قد تمَّ تصميمه؛ للتأكيد على أن كلَّ جدول في النموذج العلاقي يجب أن يحتوي على مفتاح رئيسي (يتكون من حقلٍ واحد أو أكثر) وبحيث يمكِّن هذا المفتاح من التمييز بـين الحالات التي يحتويها الجدول (أيْ: السجلات أو الصفوف) في أية حالة من الحالات التي من الممكن أن يكون عليها. كما أن قيمة المفتاح الرئيسي لأيِّ سجل يجب أن تكون صحيحةً.

وبشكلٍ خاص؛ يجب ألا تكون قيمة أيِّ حقل من الحقول المكوِّنة للمفتاح الرئيسي ذي قيمةٍ غائبة (NULL Value). وسبب ذلك يرجع إلى أن المفتاح الرئيسي يجب أن يُمكِّن من التعرُّف على السجلات المختلفة في الجدول بشكلٍ منفردٍ (دون تكرار)، وأن غياب قيمة أحد حقول المفتاح الرئيسي لن يمكِّننا من التمييز بين سجلات الجدول بشكلٍ منفرد. فعلى سبيل المثال: لو كان المفتاحُ الرئيسي (أو أحد حقوله) ذا قيمةٍ غائبة في أكثر من سجلٍ واحدٍ من سجلات جدولٍ ما؛ فإننا لن نستطيع التمييز بين هذه السجلات بشكلٍ منفرد.

3-4-1-1-4 قيود القيم الغائبة (**NULL Constraints**):

في بعض الأحيان قد لا يمكِن وضع قيمة محددة في حقلٍ ما. وقد يكون من أسباب ذلك كون الحقل لا ينطبق على الحالة التي نرغب في إدراجها ضِمن جدولٍ ما. فعلى سبيل المثال: قد يترك حقل «رقم الهاتف» في سِجل أحد أعضاء هيئة التدريس فارغاً لكون عضو هيئة التدريس الذي نحاول إدراج سجل له ضِمن سجلات أعضاء هيئة التدريس ليس لديه خط هاتفي. في مثل هذه الحالة لا ينطبق هذا الحقل على عضو هيئة التدريس؛ ولذلك يترك فارغاً للدلالة على هذا الوضع. ومن الأسباب الأخرى التي تستدعي ترك حقلٍ ما فارغاً (دون تدوين قيمة مُحدَّدة فيه) هو عندما تكون قيمة الحقل موجودةً على أرض الواقع؛ ولكنها غير متوافرة وقت إدخال السجل. فعلى سبيل المثال: قد يتمُّ إدخال سجلٍّ لعضو هيئة التدريس دون معرفة تاريخ ميلاده ليس لأن عضو هيئة التدريس ليس لديه تاريخ ميلاد؛ ولكن لكون تاريخ الميلاد غير متوفر وقت إدخال سجل عضو هيئة التدريس. في مثل هذه الحالة يتمُّ ترك حقل تاريخ الميلاد فارغاً؛ للدلالة على هذا الوضع.

ويُمكِّن النموذج العلاقي من ترك بعض الحقول فارغةً دون إدخال قِيم مُحدَّدة فيها (NULL Values) سواءً بشكلٍ ضِمني؛ من خلال تركها فارغةً (دون إدخال قيمة فيها)، أو من خلال إدخال كلمة "NULL" بشكلٍ صريح. وبذلك؛ فإن إدخال كلمة (NULL) في حقلٍ معينٍ تدلُّ على «غياب قيمة» الحقل. وغياب القيمة من حقلٍ ما لا يماثل العدد صفر أو سلسلة حرفية فارغة (Blank String)؛ لأن كلاً من العدد صفر والسلسلة الحرفية الفارغة تُعَدُّ قيماً. وعلى الرغم من أهمية مبدأ القيم الغائبة في تمثيل البيانات في النموذج العلاقي؛ فإنها تُعَدُّ مصدراً من مصادر الالتباس، كما سنوضِّح في الفصل السابع (المتعلق بلغة الاستفسار البنائية)؛ وذلك لكون القيمة الغائبة

لا يمكن مقارنتها سواءً أكان منطقياً أو حسابياً مع قيمٍ موجودة أو أخرى غائبة دون استخدام منطقٍ حسابي مطور يُمكِّن من التعامل معها.

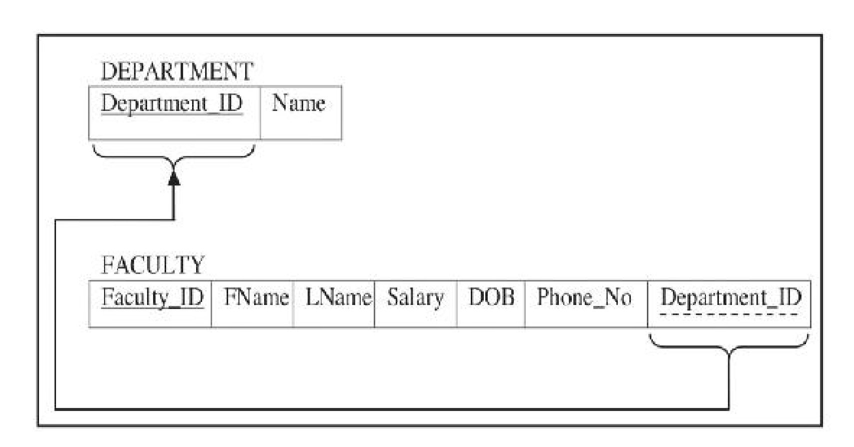
ويفيد استخدامُ قيد القيم الغائبة في تقييد القيم التي من الممكن أن يأخذها الحقل؛ بحيث إنه قد يَسْمَح بأن تكون قيمة الحقل غائبة، أو أنه قد لا يسمح بذلك. فعلى سبيل المثال: لا يمكِن أن يُدرج سجلٌ لأحد أعضاء هيئة التدريس، في جدول أعضاء هيئة التدريس، دون اسم أول لعضو هيئة التدريس واسم لعائلته. في مثل هذه الحالة يمكِن تقييدُ مثل هذين الحقلين (وهما: الاسم الأول، واسم العائلة)؛ بحيث لا يمكِن أن تكون أيٌّ من قِيمهما غائبةً من خلال وضع القيد "NOT NULL" على كلٍّ منهما.

4-1-1-4-4 قيود السلامة المرجعية (Referential Integrity Constraints):

إن قيدَ وجود مِفتاح رئيسي لكلِّ جدول في النموذج العلاقي، وقيد تكامل الجدول (من حيث وجوب أن تكون قيم المفتاح الرئيسي معرفةً دائماً) تنطبق على كلِّ جدول على حِدة. على النقيض من ذلك؛ فإن قيود السلامة المرجعية تتعلَّق بعملية الربط بين جدولين. وتضمن قيود السلامة المرجعية المحافظة على تناسق البيانات بين السجلات التابعة للجدولين. لذلك؛ فإن قيد السلامة المرجعية ينصُّ على أن أيَّ سجلٍ في أحد الجدولين يشير إلى سجلٍ في الجدول الآخر؛ فإن القيمة التي يحتويها السجل وتشير لسجلٍّ في الجدول الآخر لا بد وأن تكون موجودةً فعلاً في الجدول الآخر، أو أن تكون غائبة.

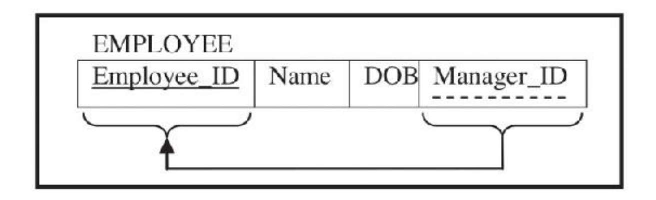
وتمثل عملية الربط بين سجلات الجدولين بما يُعرَف «المفتاح الخارجي» (Foreign Key). وعند تعريف مفتاح خارجي في جدولٍ ما؛ فإن الحقل (أو الحقول) المكوِّنة للمفتاح الخارجي لا تُعَدُّ جزءاً من خصائص (أو بيانات) الحالة التي يمثلها السجلُ نفسه؛ ولكنها جزءٌ من خصائص (أو بيانات) حالةٍ أخرى سواءٌ في نفس الجدول أو في جدول آخر. وتكون قيمة المفتاح الخارجي في السجل؛ هي قيمةٌ لأحد المفاتيح الرئيسية لسجل آخر سواء أكان في نفس الجدول أو في جدولٍ آخر. فعلى سبيل المثال: تمثل العلاقة «يعمل في» (Works_for) التي تربط بين كلِّ عضو هيئة تدريس والقسم الذي يتبعه (أو يعمل فيه) عضو هيئة التدريس؛ من خلال إدراج حقلٍ جديدٍ داخل جدول أعضاء هيئة التدريس؛ بحيث يمثل الحقل الجديد مفتاحاً خارجياً يشير إلى سجل القسم الذي يتبعه عضو هيئة التدريس. ولكون المفتاح الرئيسي في أيِّ جدولٍ هو المميزُ بين السجلات المختلفة في

الجدول؛ فإن الحقل الذي تمَّت إضافته ليصبح مفتاحاً خارجياً هو المفتاح الرئيسي لجدول الأقسام الدراسية، كما يوضح الشكل رقم (4-2).



شكل رقم (4–2): تمثيل المفاتيح الخارجية في النموذج العلاقي.

ويُقصَد بالخط المتقطع تحت «رمز القسم» (Department_ID) في جدول أعضاء هيئة التدريس؛ أن هذا الحقل عبارة عن مفتاح خارجي يشيرُ إلى حقل «رمز القسم» في جدول الأقسام الدراسية، كما هو مُوضَّح بالسهم الذي يصل بين الحقلين. وباستخدام مبدأ المفاتيح الخارجية يمكن الربط بين الجداول المختلفة في قواعد البيانات العلاقية. ويوضِّح الشكل رقم (4-3) عملية رَبط سجلات الجدول مع بعضها لتمثيل العلاقة «يدير» (Manages) التي تنصُّ على أن «الموظف الواحد يرأسه مديرٌ واحدٌ فقط، في حين أن الموظف الواحد قد يرأس صفراً أو أكثر من الموظفين». ولتمثيل هذه العلاقة يتمُّ إضافة حقل جديد ضمن جدول الموظفين؛ بحيث يحتوي هذا الحقل على قيمةٍ تمثل المفتاح الرئيسي لمدير الموظف.



شكل رقم (4–3): استخدام المفتاح الخارجي لربط سجلات الجدول نفسه ببعضها.

وعند تمثيل العلاقات (حسب تعريفها في نموذج «كينونة - علاقة»)؛ من خلال المفاتيح الخارجية؛ فإن قيود السلامة المرجعية تنصُّ على ما يلي:

1- يجب أن تُعرف حقول المفتاح الخارجي؛ بحيث تكون من نفس مجال المفتاح الرئيسي للجدول الذي يشيرُ إليه المفتاح الخارجي.

2- قيمة المفتاح الخارجي في أيِّ سجل في الجدول يجب أن تشير إلى قيمة موجودة فعلاً في الجدول الذي يشير إليه المفتاح الخارجي، أو أن تكون قيمة المفتاح الخارجي غائبة (NULL).

ففي المثال الأول أعلاه؛ يجب أن يكون حقل المفتاح الخارجي، وهو «رمز القسم الدراسي» في جدول أعضاء هيئة التدريس، من نفس مجال قيم المفتاح الرئيسي، وهو أيضاً «رمز القسم الدراسي»، في جدول أعضاء هيئة التدريس. كما يجب أن تكون كلُّ قيمة مُدوَّنة في حقل المفتاح الخارجي لكافة السجلات في جدول أعضاء هيئة التدريس - لها ما يقابلها من مفاتيح رئيسية في جدول الأقسام العلمية، أو أن تكون قيمها غائبة. أمَّا في المثال الثاني؛ فإنه يجب أن يكون حقلُ المفتاح الخارجي، وهو «رمز المدير»، من نفس مجال قيم المفتاح الرئيسي، وهو «رمز الموظف»، في الجدول نفسه. كما يجب أن تكون كلُّ قيمةٍ مدونةٍ في حقل المفتاح الخارجي لكافة السجلات في جدول الموظفين - لها ما يقابلها من مفاتيح رئيسية في نفس الجدول، أو أن تكون قيمها غائبة. ويُلاحَظ من المثال الثاني أنه ليس من الضروري أن تكون مُسمَّيات حقول المفاتيح الخارجية متطابقةً مع مُسمَّيات المفاتيح الرئيسية ما دامت من نفس المجال، وأن القيم التي تأخذها إما أن تكون موجودةً ضِمن قيم المفاتيح الرئيسية، أو أن تكون غائبة، حسب شَرطَيْ قيود السلامة المرجعية أعلاه.

والسؤال الذي قد يُطرَح هو: متى يمكِن أن يُسْمَح بأن تكون قيمُ المفتاح الخارجي غائبةً؟ ومتى لا يُسْمَح بذلك؟ إن الإجابة عن هذا التساؤل تعود بنا إلى مفهوم التعدُّدية في العلاقات (Cardinality Constraints) التي سبق أن تمَّ شرحها في الفصل الثاني. فعندما تكون العلاقة إجبارية (سواءٌ إجباري واحد، أو إجباري متعدد)، بمعنى أن القيمة الدنيا للتعدُّدية هي واحدٌ؛ فإن المفتاح الخارجي لا يمكن أن تكون قيمته غائبة. ففي المثال الأول أعلاه، لا يمكن أن تكون قيمة حقل المفتاح الخارجي غائبةً لأن كلَّ عضو هيئة تدريس "يجب" أن يعمل في قسمٍ دراسي. أمَّا في المثال الثاني، وعلى افتراض أن رئيس المنظمة لا يرأسه أحدٌ؛ فإن كلَّ موظف يجب أن يرتبط بمدير له تكون قيمة مفتاحه الرئيسي ضِمن حقل المفتاح الخارجي للموظف ما عدا رئيس المنظمة الذي ستكون قيمة حقل المفتاح الخارجي في سجلِّه غائبةً. ففي مثل هذه الحالة؛ يمكن أن تكون قيمة حقل المفتاح الخارجي

غائبة. وبناءً على إمكانية أن تكون قيمة أيِّ مفتاح خارجي غائبةً من عدم إمكانية ذلك؛ فإنه يمكن توصيف ذلك في أثناء عملية إنشاء قاعدة البيانات (كما هو مُوضَّح في الفصل السابع).

4-1-1-4-5 التعامل مع اختراق القيود في أثناء عمليات التعديل على قاعدة البيانات:

يُركِّز هذا الجزء على عمليات التعديل على قاعدة البيانات، وعلى إمكانية اختراق هذه العمليات للقيود المفروضة على قاعدة البيانات، وعلى الطرق التي تتعامل معها نظم إدارة قواعد البيانات للمحافظة على سلامة البيانات وتكاملها. ويوفر النموذج العلاقي ثلاث عمليات تعديل، هي: عملية الإضافة (Insert Operation)، وعملية الحذف (Delete Operation)، وعملية التحديث (Update Operation). وتُستخدَم عملية الإضافة لإضافة سجل أو مجموعة من السجلات لجدولٍ ما، في حين تُستخدَم عملية الحذف لحذف سجل أو أكثر من سجلات الجدول. أمَّا عملية التحديث؛ فتُستخدَم لتغيير قِيَم بعض الحقول في سجلات الجدول. وعندما تستخدم هذه العمليات؛ فإنه من الضروري التأكد من عدم اختراق أيٍّ من القيود المفروضة على هيكل قاعدة البيانات العلاقية. وفيما يلي شرحٌ للقيود التي من الممكن أن يتمَّ اختراقها عند إجراء كلٍّ من عمليات التعديل الثلاث والطرق التي من الممكن أن تُتخَذ عند محاولة اختراق أيٍّ من القيود المفروضة على هيكل قاعدة البيانات.

4-1-1-4-5-1 عملية الإضافة (The Insert Operation):

تزوِّد عملية الإضافة مجموعةً من القيم لقائمةٍ من حقول سجلٍ جديدٍ بغية إضافته إلى جدولٍ ما. ومن الممكن أن تَخْتَرِق عملية الإضافة كلاً من قيود التكامل الأربعة التي تم شرحها أعلاه. فمن الممكن أن تكون القيمة المُعطاة لأيٍّ من حقول السجل الذي ستتم إضافته غير متوافقة مع مجال الحقل. كذلك هو الحال بالنسبة لقيد تكامل الجدول؛ إذ إن السجل الجديد قد يحتوي على مفتاح رئيسي يتكرَّر مع سجل موجود أصلاً ضمن الجدول، أو أن المفتاح الرئيسي للسجل الجديد ذو قيمةٍ غائبة. كذلك هو الحال بالنسبة لقيد القيم الغائبة؛ فقيمة أحد الحقول قد تكون غائبةً على الرغم من أن القيد المفروض على الحقل الذي سيأخذ هذه القيمة لا يسمح بأن تكون القيمة غائبة. أما فيما يتعلق بقيد السلامة المرجعية؛ فإنه قد يتمُّ اختراقه إذا احتوى السجل الجديد على قيمة للمفتاح الخارجي ليس لها ما يقابلها من مفتاح رئيسي في الجدول الذي من المفترض أن يشير إليه حقل المفتاح الخارجي، أو أن تكون قيمة المفتاح الخارجي غائبةً (NULL) على الرغم من أن المفتاح الخارجي قد تمَّ توصيفه

على أساس أنه لا يمكن أن يكون ذا قيمة غائبة. وفيما يلي بعض الأمثلة لمثل هذه الاختراقات وردود فعل نظام إدارة قاعدة البيانات حيالها:

1- إضافة سجل عضو هيئة تدريس جديد يحتوي على قيمة غائبة في حقل المفتاح الرئيسي: سيقوم النظام برفض عملية الإضافة هذه لاختراقها لقيد تكامل الجدول الذي ينصُّ على أن قيمة المفتاح الرئيسي لأيِّ سجل لا يمكن أن تكون غائبةً.

2- إضافة سجل عضو هيئة تدريس يحتوي على قيمة في حقل المفتاح الخارجي الذي يربطه بالقسم الدراسي الذي يعمل فيه، وأن هذه القيمة لا تُوجَد أصلاً ضمن المفاتيح الرئيسية لجدول الأقسام العلمية؛ سيقوم النظام برفض عملية الإضافة هذه لاختراقها لقيد السلامة المرجعية.

3- إضافة سجل عضو هيئة تدريس، وقيمة حقل المفتاح الخارجي الذي يربطه بالقسم الدراسي الذي يعمل فيه غائبة (NULL)؛ سيقوم النظام برفض عملية الإضافة هذه لاختراقها لقيد القيم الغائبة؛ إذ إن كلَّ عضو هيئة تدريس "يجب" أن يرتبط بقسم يعمل فيه.

4- إضافة سجل عضو هيئة تدريس يحتوي على قيمة مكوَّنة من سلسلة حرفية ذات عشرة حروف في حقل رقم الهاتف؛ سيقوم النظام برفض عملية الإضافة هذه؛ لاختراقها قيد المجال إذا كان حقل رقم الهاتف معرفاً على أساس كونه سلسلةً حرفيةً بطول ثمانية حروف.

2-5-4-1-1-4 عملية الحذف (The Delete Operation):

من الممكن أن تخترق عملية الحذف قيد التكامل المرجعي فقط؛ وذلك إذا كان السجل المزمع حذفه مشاراً إليه من قِبل سجلات أخرى ضِمن حقول مفاتيحها الخارجية. فعلى سبيل المثال: سيتم اختراق قيد السلامة المرجعية عند محاولة حذف أيِّ سجل من سجلات جدول الأقسام الدراسية؛ وذلك عندما يكون هناك أعضاء هيئة تدريس يعملون في القسم الدراسي المزمع حذف سجله؛ لأن قيمة المفاتيح الخارجية في سجلات أعضاء هيئة التدريس ستكون مساويةً للمفتاح الرئيسي للقسم الذي سيتمُّ حذف سجله. ولو افترضنا إمكانية حذف مثل هذا السجل؛ فإن سجلات أعضاء هيئة التدريس التابعين للقسم الدراسي ستحتوي على قيم في حقول مفاتيحها الخارجية غير موجودة ضِمن المفاتيح الرئيسية لجدول الأقسام العلمية؛ مما يُعَد اختراقاً لقيد السلامة المرجعية.

يوفر النموذج العلاقي أربعة خيارات عند حدوث خروقات لقيد السلامة المرجعية؛ إذ يتمُّ استخدام الخيار المناسب عند توصيف حقل المفتاح الخارجي في أثناء إنشاء الجدول. وسيتمُّ شرح طريقة توصيف هذه الخيارات باستخدام لغة الاستفسار البنائية في الفصل السابع. أما هذه الخيارات الأربعة فهي، كما يلي:

1- رفض عملية الحذف، وهذه الطريقة تُسمَّى عملية «الرفض» (RejeCt) وهي الحالة الافتراضية عند اختراق قيد التكامل المرجعي نتيجةً لعملية الحذف. فعلى سبيل المثال: لو تمَّ تعريف قيد التكامل المرجعي لحقل المفتاح الخارجي في جدولِ أعضاء هيئة التدريس الذي يشير إلى جدول الأقسام العلمية على أنه من نوع الرفض عند اختراق قيد التكامل المرجعي؛ فإن حذف أيِّ سجلٍ من سجلات جدول الأقسام العلمية سيترتب عليه رفضُ عملية الحذف إذا وُجِد أيُّ سجلٍ من سجلات أعضاء هيئة التدريس يشيرُ إلى سجل جدول الأقسام العلمية المزمع حذفه.

2- قبول عملية الحذف مع حذف كافة السجلات التي تشيرُ للسجل الذي تمَّ حذفه، وتسمَّى هذه الطريقة «الحذف المتسلسل» (CasCade). فعلى سبيل المثال: لو تم تعريف قيد التكامل المرجعي لحقل المفتاح الخارجي في جدول أعضاء هيئة التدريس على أنه من نوع الحذف المتسلسل؛ فإن حذف أيِّ سجل من سجلات جدول الأقسام العلمية سيترتب عليه حذف كافة سجلات أعضاء هيئة التدريس التابعين لهذا القسم الدراسي.

3- قبول عملية الحذف مع تغييب قِيم الحقول التي تشير للسجل المحذوف، وتُسمَّى هذه الطريقة «تغييب القيم» (Set to NULL). فعلى سبيل المثال: لو تمَّ تعريف قيد التكامل المرجعي لحقل المفتاح الخارجي في جدول أعضاء هيئة التدريس على أنه من نوع «تغييب القيم»؛ فإن حذف أيِّ سجلٍ من سجلات جدول الأقسام العلمية سيترتب عليه تغييب القيم في حقل المفتاح الخارجي لسجلات أعضاء هيئة التدريس الذين يتبعون للقسم الدراسي الذي تمَّ حذف سجله. وتجدر الملاحظة إلى أنه في مثل هذه الحالة لا يمكن توصيف حقل المفتاح الخارجي في جدول أعضاء هيئة التدريس بطريقة لا تُمكِّن من «تغييب القيم» (NOT NULL)؛ لأن خيار الحذف هذا سيترتب عليه خرق قيد القيم الغائبة.

4- قبول عملية الحذف مع وَضْع «قيمة افتراضية» (Default Value) في الحقول التي تشير للسجل المحذوف، وتُسمَّى هذه الطريقة وضع قيمة افتراضية. فعلى سبيل المثال: لو تمَّ

تعريف قيد التكامل المرجعي لحقل المفتاح الخارجي في جدول أعضاء هيئة التدريس على أنه من نوع وَضْع قيمة افتراضية؛ فإن حذف أيِّ سجل من سجلات الأقسام العلمية سيترتب عليه، وفق هذا الخيار، وَضْع القيمة الافتراضية المصاحبة لحقل المفتاح الخارجي في كلِّ سجل من سجلات أعضاء هيئة التدريس التابعين للقسم العلمي المزمع حذف سجله.

ويُمكِّن النموذجُ العلاقي من استخدام أيِّ توليفات من الخيارات الأربعة السابقة. فعلى سبيل المثال: يُمكِن استخدام الخيار الثالث عند توصيف حقل المفتاح الخارجي في جدول أعضاء هيئة التدريس الذي يشيرُ لجدول الأقسام العلمية، في حين يمكن استخدام الخيار الثاني عند توصيف حقل المفتاح الخارجي في جدول المواد الدراسية الذي يربط كلَّ مادة دراسية بالقسم الدراسي المسئول عن تنفيذ المادة الدراسية. وعند حذف أيِّ سجل من جدول الأقسام الدراسية سيتمُّ تغييب قِيَم حقل المفتاح الخارجي الذي يشيرُ إلى جدول الأقسام الدراسية لكافة سجلات أعضاء هيئة التدريس الذين يعملون في القسم الدراسي المحذوف. أمَّا بالنسبة للمواد الدراسية التي يتمُّ تنفيذها من قِبَل القسم الذي تمَّ حذفه فسيتمُّ حذفها أيضاً، وفقاً للخيار الثاني.

4-1-1-4-5-3 عملية التحديث (**The Update Operation**):

تُستخدَم عملية التحديث لتغيير قِيم حقل أو أكثر في سجل واحدٍ أو أكثر من سجلات جدولٍ ما. وكما هو الحال بالنسبة لعملية الإضافة؛ فإن عملية التحديث من الممكن أن تخترق أياً من القيود الأربعة التي تمَّ استعراضها أعلاه. فمن الممكِن أن تكون قيمة الحقل (أو الحقول) قيد التحديث تخرج عن مجال الحقل (أو الحقول). كذلك هو الحال بالنسبة لقيد تكامل الجدول؛ إذ إنه قد تتمُّ محاولة تحديث سجلٍّ ما بحيث تكون قيمة مفتاحه الرئيسي متكررةً مع سجلٍّ آخر في نفس الجدول أو أن تكون قيمة المفتاح الرئيسي للسجل (أو جزءٍ منه) غائبةً؛ وبذلك يتمُّ اختراق قيد تكامل الجدول. كذلك هو الحال بالنسبة لقيد القيم الغائبة؛ إذ إنَّ قيمة أحد الحقول قد يتمُّ تحديثها بحيث تكون غائبةً على الرغم من أن القيد المفروض على الحقل الذي سيأخذ هذه القيمة لا يسمح بأن تكون القيمة غائبةً؛ مما يمثل اختراقاً لقيد القيم الغائبة. أمَّا فيما يتعلق بقيد السلامة المرجعية؛ فإنه قد يتمُّ اختراقه إذا تمَّ تحديث قيمة حقل المفتاح الخارجي لأحد السجلات؛ بحيث يشيرُ إلى قيمة لا يُوجَد ما يقابلها من مفتاح رئيسي في الجدول الذي من المفترض أن يشير إليه حقل المفتاح الخارجي، أو أن تكون قيمة المفتاح الخارجي غائبةً بالرغم من أن المفتاح الخارجي قد تمَّ توصيفه على أساس أنه لا يمكن أن يكون ذا

قيمة غائبـة. وفيما يلي بعض الأمثلة لمثل هذه الاختراقات وردود فعل نظام إدارة قاعدة البيانات حيالها:

1- تحديث سِجل عضو هيئة تدريس؛ بحيث تكون قيمة مفتاحه الرئيسي (أو جزء منه) غائبةً؛ سيقوم النظام برفض عملية التحديث هذه لاختراقها لقيد تكامل الجدول الذي ينصُّ على أن قيمة المفتاح الرئيسي لأيِّ سجل لا يمكن أن تكون غائبة.

2- تحديث سجلِّ عضو هيئة تدريس؛ بحيث تكون قيمة حقل مفتاحه الخارجي الجديدة التي تربطه بـالقسم الدراسي الذي يعمل فيه عضو هيئة التدريس غير موجودة أصلاً ضِمن المفاتيح الرئيسية لجدول الأقسام العلمية؛ سيقوم النظام برفض عملية التحديث هذه لاختراقها لقيد السلامة المرجعية.

3- تحديث سجل عضو هيئة تدريس؛ بحيث تكون قيمة مفتاحه الخارجي الذي يربطه بجدول الأقسام العلمية «غائبةً» (NULL)؛ سيقوم النظام برفض عملية الإضافة هذه لاختراقها لقيد القيم الغائبة؛ إذ إن كلَّ عضو هيئة تدريس "يجب" أن يرتبط بـقسمٍ يعمل فيه.

4- تحديث سجل عضو هيئة تدريس؛ بحيث تكون قيمة حقل رقم الهاتف مكونةً من سلسلة حرفية ذات عشرة حروف؛ سيقوم النظام برفض عملية التحديث هذه؛ لاختراقها لقيد المجال إذا كان حقلُ رقم الهاتف معرفاً على أساس كونه سلسلةً حرفيةً بطول ثمانية حروف.

ولإمكان تحديث قيمة المفتاح الرئيسي لأيِّ سجل في قاعدة البيانات؛ فإن مثل هذا التحديث قد يؤدي إلى اختراق قيد السلامة المرجعية؛ وذلك لأنه بالإمكان أن تُوجَد بعض السِّجلات في جداول أخرى من جداول قاعدة البيانات تشيرُ إلى المفتاح الرئيسي قبـل تحديثه. ويُمكِن تصوُّر عملية التحديث هذه، وكأنها عملية حذف للسجل تتبعها عملية إضافة لنفس السجل؛ ولكن بـمفتاح رئيسي مختلف. لذا؛ فإن الخروقات التي قد تسببها عملية التحديث هذه شبيهةٌ بالخروقات التي تسببها عملية الحذف التي سبـق إيضاحها أعلاه. لذلك؛ فإن النموذج العلاقي يوفر أربعة خيارات عند حدوث خروقات لقيد السلامة المرجعية نتيجة لعمليات التحديث، شبيهة بـتلك التي يوفرها لعمليات الحذف. ويتمُّ استخدام الخيار المناسب من الخيارات الأربـعة عند توصيف حقل المفتاح الخارجي في أثناء

إنشاء الجدول. وسيتمُّ شرح طريقة توصيف هذه الخيارات باستخدام لغة الاستفسار البنائية في الفصل السابع. أما هذه الخيارات الأربعة فهي، كما يلي:

1- رفض عملية التحديث، وهذه الطريقة تُسمَّى عملية «الرفض» (RejeCt)، وهي الحالة الافتراضية عند اختراق قيد التكامل المرجعي نتيجةً لعملية تحديث. فعلى سبيل المثال: لو تمَّ تعريف قَيْد التكامل المرجعي لحقل المفتاح الخارجي في جدول أعضاء هيئة التدريس الذي يشيرُ إلى جدول الأقسام العلمية على أنه من نوع الرفض؛ فإن تحديث قيمة المفتاح الرئيسي لأيِّ سجلٍّ من سِجلات جدول الأقسام العلمية سيترتب عليه رفض عملية التحديث إذا وُجِدَ أيُّ سجلٍّ من سجلات أعضاء هيئة التدريس يشيرُ إلى سجل جدول الأقسام العِلمية المزمع تحديث قيمة مفتاحه الرئيسي.

2- قبول عملية التحديث مع تحديث كافة حقول المفاتيح الخارجية في كافة السِّجلات التي تشير للمفتاح الرئيسي قيد التحديث. وتُسمَّى هذه الطريقة «التحديث المتسلسل» (CasCade). فعلى سبيل المثال: لو تمَّ تعريف قيد التكامل المرجعي لحقل المفتاح الخارجي في جدول أعضاء هيئة التدريس على أنه من نوع التحديث المتسلسل؛ فإن تحديث أيِّ سجل من سجلات جدول الأقسام العلمية سيترتب عليه تحديث كافة سجلات أعضاء هيئة التدريس التابعين لهذا القسم الدراسي.

3- قبول عملية التحديث مع تغييب قيم الحقول التي تشيرُ للسجل الذي تمَّ تحديث مفتاحه الرئيسي. وتُسمَّى هذه الطريقة «تغييب القيمة» (Set to NULL). فعلى سبيل المثال: لو تمَّ تعريف قيد التكامل المرجعي لحقل المفتاح الخارجي في جدول أعضاء هيئة التدريس على أنه من نوع «تغييب القيمة»؛ فإن تحديث المفتاح الرئيسي لأيِّ سجل من سجلات جدول الأقسام العلمية سيترتب عليه تغييب القيم في حقل المفتاح الخارجي لسجلات أعضاء هيئة التدريس الذين يتبعون للقسم الدراسي الذي تمَّ تحديث قيمة مفتاحه الرئيسي. وكما هو الحال في عملية الحذف، تجدر الملاحظة إلى أنه في مثل هذه الحالة لا يمكن توصيف حقل المفتاح الخارجي في جدول أعضاء هيئة التدريس بطريقة لا تمكِّن من «تغييب القيم» (NOT NULL)؛ لأن خيار التحديث هذا سيترتب عليه خرق قيد القيم الغائبة.

4- قبول عملية التحديث مع وضع «قيمة افتراضية» (Default Value) في الحقول التي تشيرُ للسِّجل الذي تمَّ تحديث مفتاحه الرئيسي. وتُسمَّى هذه الطريقة وَضْع قيمة افتراضية. فعلى سبيل المثال: لو تمَّ تعريف قيد التكامل المرجعي لحقل المفتاح الخارجي في جدول أعضاء هيئة

التدريس على أنه من نوع وَضْع قيمة افتراضية؛ فإن تحديث قيمة المفتاح الرئيسي لأيِّ سجل من سجلات الأقسام العلمية سيترتب عليه، وفق هذا الخيار، وَضْع القيمة الافتراضية المصاحبة لحقل المفتاح الخارجي في كلِّ سجل من سجلات أعضاء هيئة التدريس التابعين للقسم الدراسي الذي تمَّ تحديث قيمة مفتاحه الرئيسي.

وتسمحُ نظم قواعد البيانات العلاقية بتوصيف ردود الفعل المناسبة إزاء خروقات قيود السلامة المرجعية؛ وذلك في أثناء توصيف حقول المفاتيح الخارجية لجداول قاعدة البيانات. كما تسمح هذه النظم بتوصيف ردود الفعل حسب نوع العملية التي أدَّت إلى خرق قيد من قيود السلامة المرجعية. فعلى سبيل المثال: عند تحديث حقل رمز القسم الدراسي قد يتمُّ اختيار التحديث المتسلسل؛ مما يعني أن ينعكس رمز القسم الدراسي الجديد على كافة سجلات أعضاء هيئة التدريس التابعين للقسم الدراسي الذي تمَّ تحديث رمزه. أما في عملية الحذف؛ فإنه قد يتمُّ اختيار الرفض مما يعني عدم تنفيذ عملية الحذف عند وجود أعضاء هيئة تدريس يتبعون للقسم الدراسي المفترض حذف سجله من جدول الأقسام الدراسية. وسيتمُّ إيضاح طريقة تعريف ردود الفعل المناسبة عند وجود قيود السلامة المرجعية في أثناء شرح لغة الاستفسار البنائية في الفصل السابع.

4-2 الجبر العلاقي (**Relational Algebra**):

يستعرض هذا الجزء من الفصل الجبر العلاقي الذي يُعَدُّ إحدى «اللغات الرسمية» (Formal Languages) للنموذج العلاقي. فبالإضافة إلى المفاهيم التي تُمكِّن من تعريف هياكل البيانات والقيود المفروضة عليها؛ لا بد أن يشتمل نموذج البيانات على مجموعة من العمليات التي تُمكِّن من التعامل مع قاعدة البيانات التي تمَّت نمذجتها. وما الجبر العلاقي، إلا مجموعةٌ من العمليات الأساسية التي تُمكِّن من التعامل مع البيانات المخزَّنة وفق النموذج العلاقي لاسترجاع البيانات وفق المواصفات التي يطلبها المتعامل مع قاعدة البيانات. وتكون نتيجة الاسترجاع علاقةً جديدةً من الممكن أن تكون ناتجةً من علاقة واحدة أو أكثر. وبذلك؛ فإن العمليات الجبرية تُنتج علاقات جديدة يمكن التعامل معها بإجراء المزيد من العمليات عليها؛ وذلك من خلال استخدام نفس العمليات التي يوفرها الجبر العلاقي. ويَنْتُج عن سلسلةٍ من العمليات الجبرية ما يُعرَف «بالتعبير الجبري العلاقي» (Relational Algebra Expression) الذي تكون نتيجته علاقة جديدة تمثل ناتج عملية الاختيار (أو الاسترجاع أو الاستفسار) المنفذ على قاعدة البيانات.

ويستمدُّ الجبر العلاقي أهميته من ثلاثة أبعاد رئيسية. الأول منهما يتمثل في أن الجبر العلاقي يوفر أساساً رسمياً لإجراء العمليات على العلاقات في النموذج العلاقي. أمَّا الثاني؛ فيتمثل في كون الجبر العلاقي يُعَدُّ أساساً لبناء الاستفسارات (Queries) وزيادة فعاليتها (Optimization) في نظم إدارة قواعد البيانات العلاقية (Management Systems Relational Database). أمَّا البعد الثالث؛ فيتمثل في أنَّ بعض المفاهيم الواردة في الجبر العلاقي قد تمَّ إدراجها ضِمن اللغة القياسية للتعامل مع قواعد البيانات العلاقية والمعروفة باسم لغة الاستفسار البنائية (SQL). ولذلك؛ فإن الجبر العلاقي يُعَدُّ مكملاً لنموذج البيانات العلاقي.

ويُمكِن تصنيف عمليات الجبر العلاقي وفق فئتين رئيسيتين. فئةٌ تحتوي على عمليات مستمدة من نظرية المجموعات الحسابية؛ وذلك لكون التعريف الرَّسمي للعلاقة في النموذج العلاقي؛ هو كونها مجموعة من الصفوف (أو السجلات)؛ وبالتالي تنطبق عليها نظرية المجموعات الحسابية. وتتكوَّن هذه الفئة من: عملية الاتحاد، وعملية التقاطع، وعملية الفرق. أمَّا الفئة الثانية من العمليات الجبرية؛ فتحتوي على عملياتٍ قد تمَّ تطويرها خصيصاً لقواعد البيانات العلاقية. وتتكون هذه الفئة من: عملية الاختيار (أو الاسترجاع)، وعملية الإسقاط، وعملية إعادة التسمية، وعملية الضرب الكرتيزي، وعملية الربط، وعملية القسمة.

وسيتمُّ استعراضُ عمليات الاختيار، والإسقاط، وإعادة التسمية أولاً؛ وذلك لكونها عمليات أحادية تُطبق على علاقة واحدة فقط. بعد ذلك؛ سيتمُّ استعراض العمليات الثنائية التي تُطبق على علاقتين، عوضاً عن علاقةٍ واحدة، وبحيث يتمُّ استعراض العمليات المُستمَدَّة من نظرية المجموعات أولاً، ومن ثم استعراض العمليات الثنائية الخاصة بالنموذج العلاقي.

4-2-1 العمليات الأحادية:

4-2-1-1 عملية الاختيار (SeleCt):

تُستخدَم عملية الاختيار (SELECT) لاسترجاع مجموعة جزئية من صفوف جدولٍ ما؛ بحيث تتحقق عليها شروطٌ مُحدَّدة. ويمكن تصوُّر عملية الاختيار على أنها مُرَشِّحٌ للصفوف يُبقِي على الصفوف التي تُحقِّق شرط الاختيار في الجدول الذي تُطبق عليه العملية. كما يمكِن تصوُّرها وكأنها عملية تقسيم أفقي للجدول ينتجُ عنه مجموعتان من الصفوف، وهما: مجموعةٌ يتحقق فيها

شرط الاختيار، ويتمُّ اختيارها من خلال تنفيذ العملية، ومجموعةٌ أخرى لا يتحقق فيها شرط الاختيار، ويتمُّ استبعادها من نتيجة العملية. فعلى سبيل المثال: لنفترض وجود العلاقة (R) التالية:

R

A	B	C
a1	b1	c1
a1	b2	c2
a2	b3	c3
a3	b4	c4
a4	b5	c5
a5	b6	c6

ولنفترض أننا نرغب في اختيار الصفوف التي يتحقق فيها شرط تساوي القيمة المخزَّنة في الحقل (A) بالقيمة (a1). في هذه الحالة تطبق عملية الاختيار، كما يلي:

$$\sigma_{A = \text{“a1”}} (R)$$

وتكون نتيجة العملية جدولاً جديداً يحتوي على الصفوف التي تحتوي على القيمة (a1) في الحقل (A)، كما يلي:

$\boldsymbol{\sigma_{A = \text{“a1”}} (R)}$

A	B	C
a1	b1	c1
a1	b2	c2

وتعني هذه النتيجة أنَّ كلاً من الصف الأول والصف الثاني في العلاقة (R) قد تمَّ اختيارهما ضِمن نتيجة العملية لانطباق شرط الاختيار، وهو تساوي الحقل (A) بالقيمة (a1). في حين لم يتم انطباق الشرط على بقية صفوف العلاقة، ولذلك لم يتم اختيارها ضمن نتيجة عملية الاختيار. ومثالاً تطبيقيًّا على قاعدة بيانات الجامعة الأهلية؛ لنفترض أننا نرغب في معرفة أعضاء هيئة

التدريس الذين يعملون في قسم الحاسب الآلي ("CS") بالجامعة. يمكِن معرفة ذلك من خلال تطبيق عملية الاختيار على جدول أعضاء هيئة التدريس (FACULTY_T)؛ بحيث يكون شرط الاختيار هو تساوي قيمة الحقل (Department_ID) بالقيمة ("CS")، كما يلي:

$$\sigma_{\text{Department_ID} = \text{"CS"}} (\text{FACULTY_T})$$

في هذه الحالة تكون نتيجة عملية الاختيار العلاقة التالية:

FACULTY_	FNAME	LNAME	PHONE_NO	SALARY	DOB	DEPART
310	Saleh	Aleesa	454-8932	30000	13-SEP-66	CS
320	Mohammed	Alhamad	454-5412	44000	13-MAY-65	CS
330	Ghanim	Alghanim	456-2234	44500	12-AUG-69	CS
340	Ibraheem	Alsaleh	454-1234	25000	20-JAN-70	CS

وبشكلٍ عام تُمثَّل عملية الاختيار وفق الصيغة التالية:

$$\sigma_{<\text{Selection Condition}>} (R)$$

بحيث يُستخدَم الرمز (σ)؛ لتمثيل عملية الاختيار ويُقرأ سيجما (Sigma). أمَّا شرط الاختيار (SeleCtion Condition)؛ فهو تعبير ثنائي (Binary Operation) يُطبق على حقول العلاقة (R)؛ بحيث إمَّا أن يتحقق الشرط وتكون قيمته صحيحة (True) أو لا يتحقق الشرط وتكون قيمته خطأ (False). ويُلاحَظ أن تطبيق شرط الاختيار يتمُّ على كلِّ صفٍّ في العلاقة على حِدَة؛ بحيث يتمُّ اختيار الصف ضِمن نتيجة الاختيار عند تحقيقه لشرط الاختيار. أمَّا في حالة عدم تحقيقه لشرط الاختيار؛ فلا يتمُّ اختياره ضِمن نتيجة عملية الاختيار. أمَّا بالنسبة للعلاقة (R)؛ فإنها بشكلٍ عامٍ تعبير جبري علاقي. وفي أبسط صورها عندما تكون اسماً لعلاقة موجودة أصلاً في قاعدة البيانات دون إجراء أية عمليات جبرية عليها كما في المثالين السابقين. وتكون نتيجة عملية الاختيار علاقة جديدة تحتوي على نفس حقول العلاقة التي تمَّ تطبيق عملية الاختيار عليها.

إنَّ التعبير الثنائي الموضَّح في الشكل العام لتعليمة الاختيار أعلاه يمكن أن يتمَّ تكوينه من خلال مجموعة من التعابير البسيطة الموضَّحة بالشكل التالي:

<Attribute Name> <Comparison Operator> <Constant Value>

ففي المثال السابق؛ كان اسم الحقل (Attribute Name) هو رمز القسم (Department_ID) وعامل المقارنة هو المساواة (=) والقيمة الثابتة (Constant Value) هي أن يكون اسم القسم الحاسب الآلي ("CS"). كذلك يمكن أن تتمَّ المقارنة بحقلٍ آخر عوضاً عن قيمة ثابتة كما يوضِّح الشكل التالي:

<Attribute1 Name> <Comparison Operator> <Attribute2 Name>

ويُقصَد باســم الحقل (Attribute Name) اســمُ أحد الحقول في العلاقة (R)، في حين يُقصَــد بعامل المقارنــة (Comparison Operator) أحدُ عوامــل المقارنة في المجموعة {=, ≠, >, ≤, <, ≥}. أمَّــا القيمــة الثابتــة؛ فيجب أن تكــون من ضِمن القيــم في مجال (Domain) الحقــل الذي ســيُطبَّق عليه شرط الاختيار. ويمكِن اســتخدام عوامل الربط المنطقية {AND, OR, NOT}؛ لتكوين شروط اختيار عامة. فعلى ســبيل المثال: لاختيار أعضاء هيئة التدريس الذين يعملون في قســم الحاسب الآلي وتزيد رواتبهم عن ٣٠,٠٠٠، أو يعملون في قســم الرياضيات وتقل رواتبهم عن ٣٠,٠٠٠؛ فيمكن كتابة عملية الاختيار، كما يلي:

$$\sigma_{(Department_ID = \text{"CS"}\ AND\ Salary > 30000)\ OR\ (Department_ID = \text{"MATH"}\ AND\ Salary < 30000)}(FACULTY_T)$$

وتكون نتيجة الاختيار، كما يلي:

FACULTY_	FNAME	LNAME	PHONE_NO	SALARY	DOB	DEPART
220	Fahad	Alhamid	456-7733	25900	07-OCT-70	MATH
320	Mohammed	Alhamad	454-5412	44000	13-MAY-65	CS
330	Ghanim	Alghanim	456-2234	44500	12-AUG-69	CS

وتُطبَّــق عوامــل المقارنة {=, ≠, >, ≤, <, ≥} عــلى الحقــول ذات المدى المرتَّــب القيم (Ordered Values)، مثــل: الأعداد، والتواريخ، والسلاســل الحرفية. أما إذا كانت الحقول

ذات مدى غير مرتّب القيم؛ فلا ينطبق عليها من عوامل المقارنة سوى عامل المساواة وعامل عــدم المســاواة: {=, ≠}. ومن أمثلة الحقول ذات المدى غــير مرتب القيم حقل «الجنس» الذي يستمد قيمه من إحدى قيمتين، هما: ذكر {Male} أو أنثى {Female}.

ويتمُّ تحديدُ نتيجة عملية الاختيار من خلال تطبيق شرط الاختيار على كلِّ صف في العلاقة (R)؛ بحيث يتمُّ استبدال قيمة الحقل مكان شرط الاختيار وعندما تكون نتيجة شرط الاختيار صحيحة (True) تتمُّ عملية اختيار الصف الذي تمَّت عليه عملية المقارنة. أما إذا كانت نتيجة شرط الاختيار خطأ (False)؛ فإنه لا يتم اختيار الصف الذي تمَّت عليه عملية الاختيار. وتكون النتيجة النهائية لعملية الاختيار؛ هي جميع الصفوف التي حققت شرط الاختيار. أما بالنسبة للعمليات المنطقية {AND, OR, NOT}؛ فلها نفس التفسيرات المعروفة في الجبر الثنائي (Boolean Algebra)، وهي كالتالي:

1- <Condition1 AND Condition2> يكون صحيحاً (True) إذا كان كلٌّ من الشرط الأول (Condition1) و الشرط الثاني (Condition2) صحيحين (True).

2- <Condition1 OR Condition2> يكون صحيحاً (True) إذا كان أيٌّ من الشرط الأول (Condition1)، أو الشرط الثاني (Condition2) صحيحاً (True).

3- <NOT Condition> يكون صحيحاً إذا كان الشرط (Condition) خطأ (False)، ويكون خطأ إذا كان الشرط (Condition) صحيحاً.

وتُعَدُّ عملية الاختيار عمليةً أحاديةً (Unary Operation)؛ لكونها تنطبـق على علاقة واحدة فقط. كما أن عملية الاختيار تطبق على كلِّ صفٍّ على حِدَة وبالتالي؛ فإنَّ شروط الاختيار لا يمكن أن تُطبـق على أكثر من صفٍّ في العلاقة في نفس الوقت. ويُلاحَظ أن درجة العلاقة التي تنتج من عملية الاختيار تكون مساويةً لدرجة العلاقة الأصلية التي طُبـقت عليها عملية الاختيار. ويعني هذا أن عدد الحقول في العلاقة الناتجة مساوٍ لعدد الحقول في العلاقة الأصلية. ويُلاحَظ أيضاً أن عملية الاختيار عمليةٌ تبادليةٌ بمعنى أن نتيجة تطبيقها المتسلسل على علاقةٍ ما لا تختلف باختلاف التسلسل الذي طُبـقت فيه. ويعني هذا أن تطبيق كلٍّ من المتسلسلتين التاليتين من عملية الاختيار على العلاقة (R) متساوٍ.

$$\sigma<Condition1>(\sigma<Condition2>(R)) = \sigma<Condition2>(\sigma<Condition1>(R))$$

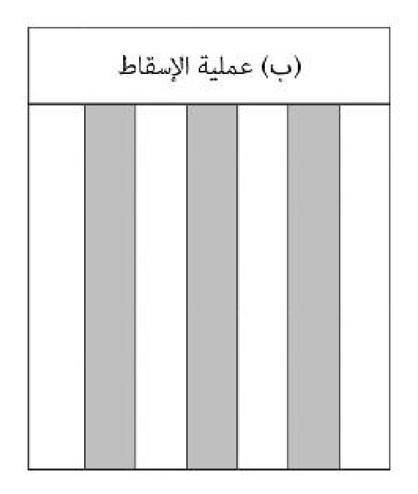
وبذلك؛ فإنه يمكن تطبيق سلسلةٍ من عمليات الاختيار على علاقةٍ ما بأيّ ترتيب كان دون إخلال بـالنتيجة النهائية للسلسلة. ونتيجةً لذلك؛ فإنه يمكن دمج سلسلة من عمليات الاختيار في عملية اختيارٍ واحدةٍ باستخدام عامل الربط الثنائي (AND)، كما يلي:

$$\sigma_{<Cond_1>}(\sigma_{<Cond_2>}(\ldots(\sigma_{<Cond_n>}(R))\ldots)) = \sigma_{<Cond_1>\ AND\ <Cond_2>\ AND\ \ldots\ AND\ <Cond_n>}(R)$$

4–2–1–2 عملية الإسقاط (**ProjeCtion**):

تُستخدَم عملية الاختيار (SELECT) لاختيار تلك الصفوف في علاقة ما بـحيث يتحقق فيها شرط الاختيار. على النقيض من ذلك؛ فإن عملية الإسقاط تختار أعمدةً معينةً من العلاقة. ففي الوقت الذي تُستخدَم فيه عملية الاختيار عند الرغبة في اختيار بعض صفوف الجدول وفق شروطٍ معينة عوضاً عن اختيارها جميعاً، والتي يمكن تصوُّرها على أنها عملية تقسيم أفقي للعلاقة؛ بحيث يتمُّ تقسيمها إلى علاقتين: علاقة تحتوي على كافة الصفوف التي تحقق شرط الاختيار، وهي التي تظهر ضِمن نتيجة الاختيار، وعلاقة تحتوي على كافة الصفوف التي لا تحقق شرط الاختيار ولا تظهر من ضمن نتائج عملية الاختيار؛ فإن عملية الإسقاط يُمكِن تصوُّرها على أنها عملية تقسيم رأسي للعلاقة؛ بـحيث ينتج عنها علاقتان: علاقة تحتوي على كافة الأعمدة التي تمَّ تحديدها ضِمن عملية الإسقاط، وهي التي تظهر ضمن نتيجة العملية، وعلاقة تحتوي على كافة الأعمدة التي لم يتم تحديدها ضِمن عملية الإسقاط ولا تظهر من ضِمن نتائج العملية. والشكل التالي يوضِّح هذا التصور لكلتا العمليتين.





فعلى سبيل المثال، لنفترض وجود العلاقة (R) التالية:

R

A	B	C
a1	b1	c1
a1	b1	c2
a2	b2	c2
a3	b3	c3
a4	b4	c4
a5	b5	c5

ولنفترض أننا نرغب في إجراء عملية إسقاط على الحقل (A) والحقل (B) من حقول العلاقة (R). في هذه الحالة تطبق عملية الإسقاط، كما يلي:

$$\pi_{A, B}(R)$$

وتكون نتيجة العملية العلاقة التالية:

$\pi_{A, B}(R)$

A	B
a1	b1
a2	b2
a3	b3
a4	b4
a5	b5

ويُلاحَظ في نتيجة عملية الإسقاط السابقة أن عدد صفوف العلاقة الناتجة منها؛ أصبح خمسة صفوف عوضاً عن الصفوف الستة التي تحتويها العلاقة الأصلية (R). ويُعزَى السبب وراء ذلك إلى أن عملية الإسقاط؛ هي عملية جبرية علاقية تُطبق على علاقات ذات خصائص مُحدَّدة وفقاً للنموذج العلاقي. وبذلك؛ فإنها يجب أن تحافظ على خصائص هذه العلاقات في نتائجها. ومن ضِمن هذه الخصائص أن العلاقة هي مجموعةٌ من الصفوف، وأن المجموعة، وفقاً لنظرية المجموعات (Set Theory)، لا تحتوي على تكرارات ضمن عناصرها. لذلك؛ فإن عملية الإسقاط، شأنها شأن بقية عمليات الجبر العلاقي؛ يجب أن تحافظ على هذه الخاصية في نتائجها. وحتى تحافظ على هذه الخاصية؛ فإن عملية الإسقاط تقوم بـالإلغاء التلقائي للصفوف المتكرِّرة من نتيجتها النهائية. وحيث إن إلغاء الحقل (C) من العلاقة (R) بعد تطبيق عملية الإسقاط يترتب عليه تكرارٌ في نتيجة الصف الأول من العلاقة مع نتيجة الصف الثاني؛ فإن عملية الإسقاط قد قامت بـإلغاء أحد الصفين من نتيجتها النهائية. وهذا هو السبب وراء حصولنا على خمسة صفوف عوضاً عن ستة صفوف في نتيجة عملية الإسقاط.

ومثالاً تطبيقياً على قاعدة بيانات الجامعة الأهلية؛ لنفترض أننا نرغب في معرفة الاسم الأول واسم العائلة والمرتب الشهري لأعضاء هيئة التدريس في الجامعة الأهلية. في هذه الحالة نستخدم عملية الإسقاط، كما يلي:

$$\pi_{FName,\ LName,\ Salary}(FACULTY_T)$$

وتكون نتيجة عملية الإسقاط هذه، كما يلي:

FNAME	LNAME	SALARY
Khalid	Aloufi	35000
Fahad	Alhamid	25900
Saleh	Aleesa	30000
Mohammed	Alhamad	44000
Ghanim	Alghanim	44500
Ibraheem	Alsaleh	25000
Ahmad	Alotaibi	33900
Saleh	Alghamdi	44600
Yahya	Khorshid	36700
Salem	Alhamad	40000
Salman	Albassam	33800
Turki	Alturki	27800
Fahad	Alzaid	44300
Saud	Alkhalifa	44900
Mahmood	Alsalem	31900
Mishal	Almazid	29800
Sultan	Aljasir	43300
Ali	Albader	45300
Saad	Alzhrani	44200
Ahmad	Alsabti	33900

ويُلاحَظ في نتيجة عملية الإسقاط السابقة أنها تمَّت بشكلٍ رأسي على الحقول التي تم تحديدها ضِمن التعليمة مُلغيةً بذلك بـقية الحقول الموجودة في العلاقة الأصلية، وأن صفوف النتيجة لا تحتوي على أية تكرارات. أمَّا الشكل العام لتعليمة الإسقاط فهو كالتالي:

$$\pi_{<\text{Attribute List}>}(R)$$

ويمثِّل الرمز (π)، ويُقرأ (باي)، تعليمة الإسقاط في الجبـر العلاقي. أما قائمة الحقول (Attribute List)؛ فتمثِّل الحقول المراد إظهارها ضِمن نتيجة عملية الإسقاط من قائمة حقول العلاقة (R). يُلاحَظ، كما هو الحال في حالة عملية الاختيار، أن العلاقة (R) عبارة عن تعبيرٍ علاقي نتيجته علاقة واحدة فقط، وأن هذا التعبير يكون في أبسط صوره عندما تكون العلاقة هي إحدى العلاقات في قاعدة البـيانات. ونتيجة عملية الإسقاط؛ هي علاقة تحتوي على الحقول المراد اختيارها فقط والمُدرَجة ضِمن قائمة الحقول في التعليمة. كما أن قائمة الحقول في النتيجة تظهر مرتـبةً وفق ترتيبـها في عملية الإسقاط. وبـهذا؛ فإن درجة العلاقة الناتجة من تعليمة الإسقاط تكون مساويةً لعدد الحقول في قائمة حقول التعليمة.

إذا كانت الحقول المُدرَجة في قائمة حقول التعليمة لا تحتوي على المفتاح الرئيسي للعلاقة (R)؛ فإنه من المحتمل أن تظهر بعض الصفوف المتكررة في نتيجة عملية الإسقاط، كما رأينا في

المثال السابق؛ إلا أن تعليمة الإسقاط تزيل هذه التكرارات عند حدوثها. لذا؛ فإن نتيجة تعليمة الإسقاط هي مجموعةٌ من الصفوف؛ مما يعني أن العلاقة الناتجة من عملية الإسقاط علاقةٌ تتوافق في مواصفاتها مع مواصفات النموذج العلاقي. تجدرُ الإشارة هنا إلى أن النموذج العلاقي الرسمي مبنيٌّ على نظرية المجموعات، كما أسلفنا سابقاً، وأن العلاقات يجب ألا تحتوي على تكرارات ضمن صفوفها؛ شأنها في ذلك شأن المجموعات الحسابية التي لا تحتوي على تكرارات ضِمن عناصرها. على النقيض من ذلك؛ فإن الحقائب (Bags) في علم الرياضيات تختلف عن المجموعات؛ بكونها قد تحتوي على تكرارات ضمن عناصرها. مثالاً تطبيقياً آخر على ذلك من قاعدة بيانات الجامعة الأهلية، لنفترض أننا نرغب في معرفة رواتب (Salary) أعضاء هيئة التدريس في الجامعة الأهلية؛ فإنه يمكن استخدام تعليمة الإسقاط التالية:

$$\pi_{Salary}(FACULTY_T)$$

وتكون نتيجة تعليمة الإسقاط السابقة، كما يلي:

```
SALARY
------
 25000
 25900
 27800
 29800
 30000
 31900
 33800
 33900
 35000
 36700
 40000
 43300
 44000
 44200
 44300
 44500
 44600
 44900
 45300
```

ويُلاحَظ في النتيجة أنها تحتوي على تسعة عشر صفاً عوضاً عن عشرين (وهي عدد أعضاء هيئة التدريس في الجامعة الأهلية). ويُعزَى السبب وراء ذلك إلى وجود عضوين من أعضاء هيئة التدريس يتقاضيان نفس الراتب، وهما: Ahmad Alotaibi، و Ahmad Alsabti، وبالتالي تمَّت

إزالة أحد الراتبين المتكررين وقدره (33.900) من نتيجة عملية الإسقاط. لذا؛ فإن عدد الصفوف الناتجة من عملية الإسقاط هي علاقةٌ تكون مساويةً أو أقل في عدد صفوفها من عدد الصفوف المدرَجة في العلاقة التي طبقت عليها عملية الإسقاط. غير أن عدد الصفوف في العلاقة الناتجة من عملية الإسقاط يكون مساوياً لعدد صفوف العلاقة التي طُبقت عليها عملية الإسقاط عندما تحتوي قائمة الحقول المدرجة ضمن حقول عملية الإسقاط على المفتاح الرئيسي للعلاقة التي طُبقت عليها عملية الإسقاط. كما أن عملية الإسقاط، وعلى النقيض من عملية الاختيار، عملية غير تبادلية، بمعنى أن نتيجة أيّ عمليتي إسقاط متعاقبتين على علاقة ما لا تكون متساوية عند اختلاف تسلسل تنفيذهما.

3-1-2-4 عملية إعادة التسمية (**Renaming**):

عند تطبيق الجبر العلاقي على علاقات قاعدة البيانات تنتج علاقات وسيطة ليس لها أسماء معينة، كما أن أسماء حقول هذه العلاقات الناتجة تكون بنفس أسماء حقول العلاقات الأصلية التي تمَّ تطبيق الجبر العلاقي عليها. وللتحكُّم في تسمية العلاقات والحقول المكونة لها يوفر الجبر العلاقي عملية إعادة التسمية. وتأخذ عملية إعادة التسمية أحد الأشكال الثلاثة التالية:

$$\pi_{Salary}\,(FACULTY_T)$$

ويُستخدَم الرمز(ρ)، ويُقرأ (روو)؛ لتمثيل عملية إعادة التسمية في جميع الأشكال الثلاثة السابقة. وتُستخدَم العملية إما لإعادة تسمية علاقة ما، أو إعادة تسمية حقول العلاقة فقط، أو إعادة تسمية كلٍّ من اسم العلاقة وحقولها معاً. فالشكل الأول للتعليمة يرمز لإعادة تسمية العلاقة (R) تحت مُسمَّى (S). وتكون نتيجة التعليمة هي كافة حقول العلاقة (R)؛ ولكن تحت مُسمَّى جديد للعلاقة باسم (S). أمَّا الشكل الثاني فيرمز إلى إعادة تسمية حقول العلاقة (R) فقط بمسميات جديدة هي (B1, B2, ..., Bn) عوضاً عن مسمياتها الأصلية. وتكون نتيجة التعليمة في هذه الحالة هي نفس العلاقة (R)؛ ولكن بمسميات حقول جديدة. ويُلاحَظ هنا أن إعادة تسمية الحقول تتمُّ بالترتيب من اليسار إلى اليمين. أمَّا الشكل الثالث للتعليمة فيرمز إلى إعادة تسمية كلٍّ من العلاقة (R) وحقولها معاً. وتكون نتيجة التعليمة في هذه الحالة علاقة بمُسمَّى جديد هو (S) تحتوي على كافة حقول العلاقة (R)؛ ولكن بمسميات جديدة. تجدر الإشارة هنا إلى أن عملية إعادة التسمية لا تغير في مسميات

العلاقات الأصلية المكونة لقاعدة البيانات أو حقولها، وإنما تقوم بإعادة تسمية هذه العلاقات أو الحقول تمهيداً لاستخدامها في عمليات جبرية علاقية أخرى. وتُعَدُّ هذه العملية مفيدةً جداً خاصةً عندما يحتوي التعبيرُ الجبري العلاقي على سلسلة طويلة من العمليات الجبرية العلاقية؛ مما قد يستدعي تجزئته حتى يمكِن التعامل معه بسهولة. فلإظهار الاسم الأول واسم العائلة والراتب لأعضاء هيئة التدريس الذين يعملون في قسم الحاسب الآلي ("CS") بالجامعة الاهلية، على سبيل المثال - تُطبق عملية اختيار وعملية إسقاط على علاقة أعضاء هيئة التدريس (FACULTY_T)، كما يلي:

$$\pi_{FName,\ LName,\ Salary}(\sigma_{Department_ID=\text{"CS"}}(FACULTY_T))$$

ويُوضِّح الشكل التالي ناتج هذا التعبير الجبري العلاقي:

FNAME	LNAME	SALARY
Saleh	Aleesa	30000
Mohammed	Alhamad	44000
Ghanim	Alghanim	44500
Ibraheem	Alsaleh	25000

وبديلاً عن كتابة التعبير الجبري العلاقي السابق الذي يحتوي على عمليتين جبريتين متداخلتين؛ يمكن كتابة التعبير على أنه سلسلةٌ من عمليتين جبريتين علاقيتين، وذلك من خلال تسمية النتائج الوسيطة، كما يلي:

$$CS_Employees \leftarrow (\sigma_{Department_ID=\text{"CS"}}(FACULTY_T)$$

$$Result \leftarrow \pi_{FName,\ LName,\ Salary}(CS_Employees)$$

ويكون ناتج العملية الأولى العلاقة (CS_Employees) الموضَّحة، كما يلي:

FACULTY_	FNAME	LNAME	PHONE_NO	SALARY	DOB	DEPART
310	Saleh	Aleesa	454-8932	30000	13-SEP-66	CS
320	Mohammed	Alhamad	454-5412	44000	13-MAY-65	CS
330	Ghanim	Alghanim	456-2234	44500	12-AUG-69	CS
340	Ibraheem	Alsaleh	454-1234	25000	20-JAN-70	CS

أمَّا ناتج العملية الثانية؛ فيكون العلاقة (Result) الموضَّحة، كما يلي:

FNAME	LNAME	SALARY
Saleh	Aleesa	30000
Mohammed	Alhamad	44000
Ghanim	Alghanim	44500
Ibraheem	Alsaleh	25000

وعادةً ما يكون تفكيك تعبير جبري علاقي واحد مكوَّناً من سلسلةٍ طويلةٍ من العمليات الجبرية، واستخراج نتائج وسيطة منها وصولاً للنتيجة النهائية - أسهل في التعامل من استخدام التعبير كوحدة واحدة. ويُمكِن استخدام عملية إعادة تسمية الحقول في النتائج الوسيطة والنتائج النهائية. وتصبح عملية إعادة التسمية ذات أهميةٍ كبيرة خاصةً عند استخدام عمليات أكثر تعقيداً، مثل: عملية الاتحاد، وعملية التقاطع اللتين سنتطرق لهما في الجزء التالي.

4–2–2 العمليات الثنائية:

4–2–2–1 عمليات الجبر العلاقي الثنائية من نظرية المجموعات الحسابية:

إن عملية الاختيار وعملية الإسقاط تمكِّنان من الحصول على معلومات معينة من خلال تطبيقهما على علاقة واحدة فقط في أيِّ وقتٍ من الأوقات؛ وذلك لكونهما عمليات أحادية لا تطبقان إلا على علاقة واحدة فقط. غير أننا نحتاج في الكثير من الأحيان إلى الحصول على معلومات لا يمكن الوصول إليها إلا من خلال عمليات تقوم بالدمج ما بين العلاقات. وتُعَدُّ هذه العمليات عمليات ثنائية؛ لكونها تتعامل (أو تُطبق) على علاقتين في نفس الوقت. ونستعرض في هذا الجزء عمليات الجبر العلاقي الثنائية المستمدَّة من نظرية المجموعات الحسابية. وهذه العمليات، هي: عملية الاتحاد، وعملية التقاطع، وعملية الفرق.

4–2–2–1–1 عملية الاتحاد (Union Operation):

ينتج من تطبيق عملية الاتحاد على علاقتين (R) و (S) علاقةٌ جديدةٌ لها نفس بنية العلاقتين اللتين تمَّ تطبيق عملية الاتحاد عليهما. وتكون نتيجة العملية علاقة جديدة تتكون من كافة الصفوف الموجودة في العلاقتين؛ ولكن دون تكرار. وتأتي هذه العملية متوافقةً مع عملية الاتحاد في نظرية المجموعات الحسابية؛ إذ إن عملية الاتحاد بين أيِّ مجموعتين من العناصر تكون مجموعة جديدة تحتوي على عناصر كلتا المجموعتين دون تكرار في عناصر المجموعة الناتجة. ولتطبيق عملية الاتحاد في الجبر العلاقي؛ فإنه يُشترَط أن تكون كلتا العلاقتين قيد تنفيذ عملية الاتحاد عليهما

متوافقتين؛ من حيث عملية الاتحاد (Union-Compatible) بمعنى أن يكون لكلتا العلاقتين نفس عدد الحقول، وأن يكون كلُّ زوج من الحقول، المتقابلين في الترتيب، في كلتا العلاقتين من نفس المجال (Domain). ويكون عددُ صفوف العلاقة الناتجة بـحَدٍّ أقصى مساوياً لحاصل جمع عدد صفوف العلاقتين. غير أن هذا العدد يتناقص في حالة وجود تكرار ما بين صفوف العلاقتين؛ لأن كلَّ صف متكرر ما بـين العلاقتين لا يظهر في النتيجة إلا مرة واحدة فقط. وتمثل عملية الاتحاد بين علاقتين (R) و (S) كما هو مُوضَّح في الشكل التالي:

$$R \cup S$$

حيث إن الرمز () يُستخدَم لتمثيل عملية الاتحاد. ويُلاحَظ أن عملية التوافق من حيث الاتحاد لا تنصُّ على أن يحمل كلُّ زوج من الحقول، المتقابلين في الترتيب، في العلاقتين نفس المُسمَّى؛ وذلك لأنه باستطاعتنا دائماً أن نعيد تسمية الحقول لتتوافق مع بـعضها من حيث المسميات. ومن الأمثلة التطبيقية لعملية الاتحاد لنفترض وجود العلاقتين (R) و(S) المتوافقتين من حيث الاتحاد، وأننا نرغب في إجراء عملية اتحاد بـينهما. فستكون نتيجة عملية الاتحاد ما بين العلاقتين، كما يلي:

R

A	B
a1	b1
a2	b2
a3	b3
a4	b4

$\cup$

S

A	B
a1	b1
a2	b2
a3	b1

$\Rightarrow$

R ∪ S

A	B
a1	b1
a2	b2
a3	b3
a3	b1
a4	b4

ويُلاحَظ في العلاقة الناتجة من عملية الاتحاد احتواؤها على خمسة صفوف عوضاً عن الصفوف السبعة التي تحتويها العلاقتان (R) و(S)؛ وذلك لتكرار الصف الأول والصف الثاني في كلتا العلاقتين. وبالتالي؛ فإن عدد الصفوف في ناتج عملية الاتحاد بـين العلاقتين أقل من حاصل جمع عدد صفوفهما. كما يُلاحَظ أن الصف الثالث في العلاقة (R) والصف الثالث في العلاقة (S) قد تمَّ إدراجهما ضِمن نتيجة العملية على الرغم من وجود تطابق جزئي بينهما؛ إذ إن القيمة المخزَّنة في الحقل (A) في كلا الصفين متساوية. ويعني هذا أن أيَّ صفين في علاقتين ما يُعدَّان متساويين عند

تساوي كافة القيم المخزَّنة في كافة حقولهما فقط. لذلك لا يُعَدُّ الصف الثالث في العلاقة (R) والصف الثالث في العلاقة (S) متطابقين. وبذلك يجب إدراجهما ضمن نتيجة عملية الاتحاد كلٌّ على حِدة.

ومن الأمثلة التطبيقية على قاعدة بيانات الجامعة الأهلية؛ لنفترض أننا نرغب في معرفة الاسم الأول واسم العائلة لكافة أعضاء هيئة التدريس وكافة الطلبة في الجامعة. في هذه الحالة نطبق عملية إسقاط على علاقة أعضاء هيئة التدريس لاختيار حقول الاسم الأول واسم العائلة. ونطبق عملية إسقاط أخرى مماثلة على علاقة الطلبة. وتكون نتائج العمليتين في هذه الحالة متوافقةً من حيث عملية الاتحاد؛ لكون كلتا العلاقتين الوسيطتين الناتجتين لهما نفس عدد الحقول (وهما حقلان يمثلان الاسم الأول واسم العائلة)، وأن كلَّ زوج من الحقول المتقابلة لهما نفس المجال؛ إذ إنهما معرفان على أساس أنهما حقول حرفية بطول (12) حرفاً ((CHAR(12)). أمَّا شكل العمليات المطبقة للحصول على النتيجة المطلوبة فهو، كما يلي:

$$(\pi_{FName, LName}(FACULTY_T)) \cup (\pi_{FName, LName}(STUDENT_T))$$

ويمكِن اعتبارُ عملية الاتحاد عمليةً متعددةً؛ بمعنى أنه يمكن تطبيقها على أيِّ عددٍ من العلاقات يفوق الاثنتين ضِمن سلسلةٍ من عمليات الاتحاد، وأنها عملية مشاركة (أو تبادلية) (AssoCiative Operation)؛ بمعنى أن ترتيب تطبيقها على عددٍ من العلاقات ضِمن سلسلة من عمليات الاتحاد لا يؤثر في النتيجة النهائية لسلسلة عمليات الاتحاد. ويمكِن تمثيل هذه الخاصية، كما يلي:

$$R \cup (S \cup T) = (R \cup S) \cup T$$

في سلسلة عمليات الاتحاد السابقة؛ تكون نتيجة تطبيق العملية على العلاقتين (S) و(T)، ومن ثم تطبيق عملية الاتحاد على العلاقة الناتجة، والعلاقة (R) (كما هو مُوضَّح في الجزء الأيسر من المعادلة) مساويةً لنتيجة تطبيق عملية الاتحاد على العلاقتين (R) و (S) ومن ثم تطبيق عملية الاتحاد على العلاقة الناتجة والعلاقة (T) (كما هو مُوضَّح في الجزء الأيمن من المعادلة).

2-1-2-2-4 عملية التقاطع (InterseCtion Operation):

ينتجُ من تطبيق عملية التقاطع على علاقتين (R) و (S)، وكما هو الحال في عملية الاتحاد؛ علاقةٌ جديدةٌ لها نفس بنية العلاقتين اللتين تم تطبيق عملية التقاطع عليهما. إلا أن ناتج عملية التقاطع، وعلى خلاف عملية الاتحاد، يتكون من الصفوف المشتركة في العلاقتين بمعنى أن كل صف من صفوف نتيجة العملية يجب أن يكون موجوداً في كلتا العلاقتين. وتأتي هذه العملية متوافقةً مع عملية التقاطع في نظرية المجموعات الحسابية؛ إذ إن عملية التقاطع بين أيَّة مجموعتين من العناصر تكون مجموعة جديدة تحتوي على العناصر المشتركة بين عناصر المجموعتين. ولتطبيق عملية التقاطع في الجبر العلاقي؛ فإنه يشترط، كما هو الحالُ في عملية الاتحاد، أن تكون العلاقتان متوافقتين من حيث عملية الاتحاد (Union-Compatible). ويعني هذا أن يكون للعلاقتين نفسُ عددِ الحقول، وأن يكون لكلِّ زوجٍ من الحقول، المتقابلين في الترتيب، في كلتا العلاقتين نفس المجال (Domain). وتُمثَّل عملية التقاطع بين علاقتين (R) و(S) كما هو مُوضَّح في الشكل التالي:

$$R \cap S$$

حيث إن الرمز (∩) يُستخدَم لتمثيل عملية التقاطع. ومن الأمثلة التطبيقية لعملية التقاطع؛ لنفترض وجود العلاقتين (R) و(S) المتوافقتين من حيث الاتحاد، وأننا نرغب في إجراء عملية تقاطع بينهما. فستكون نتيجة عملية التقاطع بين العلاقتين، كما يلي:

R

A	B
a1	b1
a2	b2
a3	b3
a4	b4

∩

S

A	B
a1	b1
a2	b2
a3	b1

⇒

R ∩ S

A	B
a1	b1
a2	b2

ويُلاحَظ في نتيجة عملية التقاطع السابقة بين العلاقتين (R) و(S) أنها تحتوي على الصفوف المشتركة بين العلاقتين، وهما: الصف الأول والصف الثاني. كما يُلاحَظ عدم إدراج أيٍّ من الصف الثالث في العلاقة (R)، أو الصف الثالث في العلاقة (S) ضِمن نتيجة العملية على الرغم من وجود

تطابقٍ جزئي بينهما؛ إذ إنَّ القيمة المخزَّنة في الحقل (A) في كلا الصفين متساوية. ويعني هذا أن أيَّ صفين في علاقتين ما يُعدَّان متساويين فقط في حال تساوت كافة القيم المخزَّنة في كافة حقولهما. لذلك لا يُعَدُّ الصف الثالث في العلاقة (R) والصف الثالث في العلاقة (S) متطابقين. وبذلك يجب عدم إدراج أيٍّ منهما ضِمن نتيجة عملية التقاطع.

ومن الأمثلة التطبيقية على قاعدة بيانات الجامعة الأهلية؛ لنفترض وجود بعض أعضاء هيئة التدريس الذين سبق أن كانوا من طلبة الجامعة قبل تعيينهم فيها أعضاءً لهيئة التدريس، وأننا نرغب في معرفة هؤلاء الأعضاء؛ من خلال إظهار الاسم الأول واسم العائلة لكلٍّ منهم. في هذه الحالة نطبق عملية إسقاط على علاقة أعضاء هيئة التدريس؛ لاختيار حقول الاسم الأول واسم العائلة. ونطبق عملية إسقاط أخرى مماثلة على علاقة الطلبة. وتكون نتائج العمليتين في هذه الحالة متوافقةً من حيث عملية الاتحاد؛ لكون كلتا العلاقتين الوسيطتين الناتجتين لهما نفس عدد الحقول (وهما حقلان يمثلان الاسم الأول واسم العائلة)، وأن كلَّ زوجٍ من الحقول المتقابلة لهما نفس المجال؛ إذ إنهما معرفان على أساس أنهما حقول حرفية بطول (12) حرفاً ((CHAR(12)). أمَّا شكل العمليات المطبقة للحصول على النتيجة المطلوبة؛ فهو كما يلي:

$$(\pi_{FName,\ LName}\ (FACULTY_T)) \cap (\pi_{FName,\ LName}\ (STUDENT_T))$$

ويمكِن اعتبارُ عملية التقاطع، وكما هو الحالُ في عملية الاتحاد، عملية متعددة بمعنى أنه يمكِن تطبيقها على أيِّ عددٍ من العلاقات يفوق الاثنتين ضِمن سلسلة من عمليات التقاطع، وأنها عملية مشاركة (أو تبادلية) (AssoCiative Operation)؛ بمعنى أن ترتيب تطبيقها على عددٍ من العلاقات ضِمن سلسلةٍ من عمليات التقاطع لا يؤثر في النتيجة النهائية لسلسلة عمليات التقاطع. ويمكِن تمثيل هذه الخاصية، كما يلي:

$$R \cap (S \cap T) = (R \cap S) \cap T$$

في سلسلة عمليات التقاطع السابقة؛ تكون نتيجة تطبيق العملية على العلاقتين (S) و(T)، ومن ثمَّ تطبيق عملية التقاطع على العلاقة الناتجة والعلاقة (R) (كما هو مُوضَّح في الجزء الأيسر من المعادلة) مساويةً لنتيجة تطبيق عملية التقاطع على العلاقتين (R) و(S) ومن ثم تطبيق عملية التقاطع على العلاقة الناتجة والعلاقة (T)، (كما هو مُوضَّح في الجزء الأيمن من المعادلة).

3-1-2-2-4 عملية الفرق (**Minus Operation**):

ينتجُ من تطبيق عملية الفرق على العلاقتين (R) و (S) - وكما هو الحالُ في عملية الاتحاد وعملية التقاطع - علاقةٌ جديدةٌ لها نفس بنية العلاقتين اللتين تمَّ تطبيق عملية الفرق عليهما. غير أن ناتج عملية الفرق، وعلى خِلاف عملية الاتحاد وعملية التقاطع، يتكون من كافة الصفوف الموجودة في العلاقة الأولى، وليست موجودةً في العلاقة الثانية. وتأتي هذه العملية متوافقةً مع عملية الفرق في نظرية المجموعات الحسابية؛ إذ إنَّ عملية الفرق بين أيِّ مجموعتين من العناصر تكون مجموعة جديدة تحتوي على كافة عناصر المجموعة الأولى، وليست موجودةً ضِمن عناصر المجموعة الثانية. ولتطبيق عملية الفرق في الجبر العلاقي؛ فإنه يشترط، كما هو الحال في عملية الاتحاد وعملية التقاطع، أن تكون كلتا العلاقتين قيد تنفيذ عملية الفرق عليهما أن يكونا متوافقتين؛ من حيث عملية الاتحاد (Union-Compatible). ويعني هذا أن يكون لكلتا العلاقتين نفسُ عددِ الحقول، وأن يكون لكلِّ زوجٍ من الحقول، المتقابلين في الترتيب، في كلتا العلاقتين نفس مجال القيم (Domain) التي يمكِن أن يحتويا عليها. وتمثِّل عملية الفرق بين علاقتين (R) و (S) كما هو مُوضَّح في الشكل التالي:

R – S

حيث إن الرمز (–) يُستخدَم لتمثيل عملية الفرق. ومن الأمثلة التطبيقية لعملية الفرق؛ لنفترض وجود العلاقتين (R) و (S) المتوافقتين؛ من حيث الاتحاد وأننا نرغب في إجراء عملية فرق بينهما. فستكون نتيجة عملية الفرق بين العلاقتين، كما يلي:

R

A	B
a1	b1
a2	b2
a3	b3
a4	b4

–

S

A	B
a1	b1
a2	b2
a3	b1

⇒

R – S

A	B
a3	b3
a4	b4

ويُلاحَظ في نتيجة عملية الفرق السابقة بين العلاقتين (R) و (S) أنها تحتوي على الصفوف الموجودة في العلاقة (R)، وليست موجودةً في العلاقة (S). وفي هذه الحالة هما الصفان الثالث والرابع من صفوف العلاقة (R).

ومن الأمثلة التطبيقية على قاعدة بيانات الجامعة الأهلية؛ لنفترض مرةً أخرى وجود بعض أعضاء هيئة التدريس الذين سبق أن كانوا من طلبة الجامعة قبل تعيينهم فيها أعضاءً لهيئة التدريس، وأننا نرغب في معرفة أعضاء هيئة التدريس الذين لم يسبق لهم أن كانوا من طلبة الجامعة؛ وذلك من خلال إظهار الاسم الأول واسم العائلة لكلٍّ منهم. في هذه الحالة نطبق عملية إسقاط على علاقة أعضاء هيئة التدريس لاختيار حقول الاسم الأول واسم العائلة. ونطبق عملية إسقاط أخرى مماثلة على علاقة الطلبة. وتكون نتائج العمليتين في هذه الحالة متوافقةً من حيث عملية الاتحاد؛ لكون كلتا العلاقتين الوسيطتين الناتجتين لهما نفس عدد الحقول (وهما حقلان يمثلان الاسم الأول واسم العائلة)، وأنَّ كل زوجٍ من الحقول المتقابلة لهما نفس المجال؛ إذ إنهما معرفان على أساس أنهما حقول حرفية بطول (12) حرفاً ((CHAR(12)). أما شكل العمليات المطبقة للحصول على النتيجة المطلوبة فهو، كما يلي:

$$(\pi_{FName,\ LName}\ (FACULTY_T)) - (\pi_{FName,\ LName}\ (STUDENT_T))$$

ويُلاحَظ على عملية الفرق، وبشكلٍ عام، وعلى خلاف عملية الاتحاد وعملية القاطع، أنها عملية غير مشاركة (أو تبادلية) بمعنى أن نتيجة تطبيقها المتسلسل يختلف باختلاف التسلسل الذي طبقت فيه. ويمكن توضيح ذلك، كما يلي:

$$R - S \neq S - R$$

ولإيضاح هذه الخاصية نذكرُ مثالاً تطبيقياً على قاعدة بيانات الجامعة الأهلية: لنفترض أننا طبقنا عملية الفرق بين نتيجة عملية الإسقاط على علاقة أعضاء هيئة التدريس ونتيجة علاقة الإسقاط على علاقة الطلبة، في المثال السابق، بشكلٍ معاكسٍ كما هو مُوضَّح فيما يلي:

$$(\pi_{FName,\ LName}\ (STUDENT_T)) - (\pi_{FName,\ LName}\ (FACULTY_T))$$

في هذه الحالة لن تكون نتيجة تطبيق عملية الفرق ممثلةً للمطلوب، وهو «إظهار أسماء أعضاء هيئة التدريس الذين سبق أن كانوا من طلبة الجامعة». وإنما ستكون النتيجة «أسماء الطلبة الذين ليسوا من أعضاء هيئة التدريس». وقد تكون مثل هذه النتيجة غير ذات مدلولات منطقية إذا ما اعتبرنا أن عضو هيئة التدريس لا يمكن أن يكون طالباً في الجامعة!

4-2-2-2 عمليات الجبر العلاقي الثنائية الخاصة بالنموذج العلاقي:

يستعرض هذا الجزءُ عمليات الجبر العلاقي الثنائية الخاصة بالنموذج العلاقي. وهذه العمليات، هي: عملية الضرب الكرتيزي، وعملية الربط، وعملية القسمة.

4-2-2-2-1 عملية الضرب الكرتيزي (Cartesian ProduCt):

عملية الضرب الكرتيزي؛ هي عمليةٌ تُطبق على المجموعات، وهي عمليةٌ ثنائيةٌ بين معطيين يكون كلٌّ منهما مجموعةً من الصفوف تتمثل في علاقةٍ ما. وتُستخدَم هذه العملية لدمج صفوف علاقتين بشكلٍ توليفي؛ بمعنى إظهار جميع الاحتمالات الممكنة؛ للدمج بين صفوف العلاقتين. وتكون نتيجة العملية التي يُرمَز لها برمز علامة الضرب (×) علاقةً جديدةً ذات درجة تساوي حاصل جمع درجة العلاقتين اللتين تمَّت عليهما عملية الضرب الكرتيزي. ونعني بدرجة العلاقة هنا، كما أسلفنا سابقاً، عدد حقول العلاقة. كما يكون عدد صفوف العلاقة الناتجة مساوياً لعددِ صفوف العلاقة الأولى مضروباً في عدد صفوف العلاقة الثانية. وتمثِّل عملية الضرب الكرتيزي بين علاقتين (R1) و (R2) كما هو موضح في الشكل التالي:

$$R_1 \times R_2$$

ومن الأمثلة التطبيقية لعملية الضرب الكرتيزي؛ لنفترض وجود العلاقتين (R1) ذات الدرجة (3) وحقولها هي (A, B, C) والعلاقة (R2) ذات الدرجة (2) وحقولها هي (Y, Z)، وأننا نرغب في إجراء عملية الضرب الكرتيزي عليهما. عندئذ ستكون نتيجة عملية الضرب الكرتيزي بين العلاقتين كما هو مُوضَّح بالشكل التالي:

R_1

A	B	C
a1	b1	c1
a2	b2	c2
a3	b3	c3

×

R_2

Y	Z
y1	z1
y2	z2

⇒

$R_1 \times R_2$

A	B	C	Y	Z
a1	b1	c1	y1	z1
a1	b1	c1	y2	z2
a2	b2	c2	y1	z1
a2	b2	c2	y2	z2
a3	b3	c3	y1	z1
a3	b3	c3	y2	z2

ويُلاحَظ أن درجة العلاقة الناتجة أصبحت (5)، وحقولها هي حقول العلاقة (R1) مدموجة مع حقول العلاقة (R2). أمَّا نتيجة العملية؛ فهي دمج للصف الأول من العلاقة (R1) مع كلِّ صفٍّ من صفوف العلاقة (R2)، ودمج للصف الثاني من العلاقة (R1) مع كل صف من صفوف العلاقة (R2)، ودمج للصف الثالث من العلاقة (R1) مع كل صف من صفوف العلاقة (R2). ويعني هذا أنَّ نتيجة الضرب الكرتيزي ما هي إلا استعراض لكافة توليفات صفوف علاقتين؛ بحيث ينتج عن هذه التوليفات علاقة جديدة درجتها هي حاصل جمع درجتي العلاقتين، وهي عددُ حقول العلاقة الأولى مجموعة على عدد حقول العلاقة الثانية، اللتين تمَّت عليهما عملية الضرب الكرتيزي. ويكون عدد صفوفها هو حاصل ضرب عدد صفوف العلاقة الأولى في عددِ صفوف العلاقة الثانية. ففي مثالنا المستخدَم أصبح عدد صفوف العلاقة الناتجة هو (6)، والذي يمثل حاصل ضرب عدد صفوف (R1)، وهو (3)، بعدد صفوف (R2)، وهو (2). وتجدر الإشارة هنا إلى أنه لا يُشترَط في الضرب الكرتيزي توافق العلاقتين قيد تنفيذ إجراء العملية عليهما؛ من حيث الاتحاد كما هو الحال في عمليات الاتحاد، والتقاطع، والفرق.

ولأنه من الممكن أن تحتوي علاقتان على نفس مسميات الحقول؛ فإنه يتمُّ إضافة مُسمَّى العلاقة للحقول المتشابهة في نتيجة عملية الضرب الكرتيزي؛ حتى تتمَّ المحافظة على خاصية تفرُّد مسميات الحقول في العلاقة الناتجة. كما أنه يمكِن استخدام عملية إعادة التسمية مع إحدى العلاقتين لإعادة تسمية حقولها المتشابهة في المُسمَّى مع العلاقة الأخرى قبل إجراء عملية الضرب الكرتيزي عليهما.

ومن الأمثلة التطبيقية على قاعدة بيانات الجامعة الأهلية؛ لنفترض أننا نرغب في إظهار أسماء أعضاء هيئة التدريس (الاسم الأول، واسم العائلة) مقروناً بـأسماء الأقسام التي يعملون فيها. في هذه الحالة يتمُّ تنفيذ التعبير الجبري العلاقي التالي:

$$\pi_{FName,\ LName,\ Department_ID}\ (\sigma_{Department_ID\ =\ FDEPT_ID}\ (\rho_{Department_ID\ as\ FDEPT_ID}\ (FACULTY_T) \times DEPARTMENT_T))$$

وتُنفَّذ العملياتُ في التعبير الجبري العلاقي السابق وفق أولويات متوافقة مع الأقواس المُستخدَمة؛ وذلك من الداخل للخارج؛ بحيث تُنفَّذ عملية إعادة التسمية لحقل رمز القسم (Department_ID) في علاقة أعضاء هيئة التدريس أولاً ليصبح (FDEPT_ID). ويُعزَى السبب وراء إعادة تسمية هذا الحقل إلى تطابق مُسمَّاه مع مُسمَّى حقل آخر في علاقة الأقسام العلمية (DEPARTMENT_T). بعد ذلك تنفذ عملية الضرب الكرتيزي الذي تكون نتيجته كافة التوليفات الممكنة ما بين صفوف علاقة أعضاء هيئة التدريس مع صفوف علاقة الأقسام العلمية. ونظراً لأننا قد قمنا بإعادة تسمية الحقل الممثل لرمز القسم في علاقة أعضاء هيئة التدريس؛ فإن العلاقة الناتجة لا تحتوي على أيَّة ازدواجية في مُسمَّيات حقولها وبالتالي؛ فإنها علاقةٌ صحيحةٌ تتوافق مع شروط علاقات النموذج العلاقي. بعد ذلك تُنفَّذ عملية الاختيار التي تقوم باختيار الصفوف التي تتساوى فيها حقول رمز القسم الذي يعمل فيه عضو هيئة التدريس مع رمز القسم المدوَّن في قائمة الأقسام العلمية. وتكون النتيجة الوسيطة حتى هذه المرحلة من التعبير علاقةً تحتوي على صفوف مكونة من كافة حقول علاقة أعضاء هيئة التدريس مربوطة بحقول الأقسام الدراسية التي يعملون فيها. وأخيراً؛ تُنفَّذ عملية الإسقاط التي تقوم بإظهار الحقول الثلاثة المطلوبة، وهي: الاسم الأول لعضو هيئة التدريس، واسم عائلته، واسم القسم الذي يتبعه عضو هيئة التدريس.

2-2-2-2-4 عملية الربط (**Join Operation**):

تُستخدَم عملية الربط (Join)؛ لدمج سجلات تربط بينهما علاقةٌ ما بحيث تتبع هذه السجلات لجدولين مختلفين. ويُرمَز لعملية الربط بالرمز (⋈). وتُعَدُّ عملية الربط واحدةً من أهمِّ العمليات في النموذج العلاقي؛ لكونها تمكِّن من معالجة العلاقات التي تربط بين الجداول المختلفة في قاعدة البيانات. ولإيضاح ذلك؛ لنفترض أننا نرغب في معرفة أسماء الطلبة وأرقام المواد الدراسية التي سجلوا فيها، ونتائجهم في هذه المواد. في هذه الحالة يتم ربط جدول التسجيل

(ENROLLMENT_T) مع جدول الطلبة (STUDENT_T)؛ للحصول على كافة بيانات الطلبة والمواد التي قاموا بالتسجيل فيها. بعد ذلك تُتبع عملية الربط بعملية إسقاط على الحقول المطلوبة، كما يلي:

$$Result \leftarrow \pi_{FName,\ LName\ Course_ID,\ Grade}\ (STUDENT_T \bowtie ENROLLMENT_T)$$

وتُسمَّى العملية السابقة عملية ربط التساوي؛ إذ إن عملية الرَّبط تتم بين جدول الطلبة، وجدول التسجيل وفق الحقول المشتركة بين الجدولين (وهي حقل «رقم الطالب» في مثالنا). ويمكن كتابة عملية ربط التساوي (Equi_Join) بشكلٍ صريح، كما يلي:

$$STUDENT_T \bowtie_{STUDENT_T.Student_ID\ =\ ENROLLMENT_T.Student_ID} ENROLLMENT_T$$

وتكون نتيجة العملية السابقة بعد إجراء عملية الإسقاط على الحقول المطلوبة، كما يلي:

FNAME	LNAME	COURSE_	GRADE
Saleh	Alhamad	CHEM101	4
Abdullah	Aloufi	CHEM101	3
Khalid	Alsultan	CHEM101	4
Salem	Algamdi	CHEM101	3
Mishal	Alyousef	CHEM101	1
Saleh	Alhamad	CS101	2
Mishal	Alyousef	CS101	4
Saleh	Alhamad	CS102	3
Mishal	Alyousef	CS102	4
Saleh	Alhamad	ENGL101	3
Abdullah	Aloufi	ENGL101	4
Salem	Algamdi	ENGL101	4
Mishal	Alyousef	ENGL101	4
Saleh	Alhamad	ENGL102	1
Mishal	Alyousef	ENGL102	4
Saleh	Alhamad	MATH101	3
Abdullah	Aloufi	MATH101	2
Salem	Algamdi	MATH101	0
Mishal	Alyousef	MATH101	2
Saleh	Alhamad	MATH102	2
Mishal	Alyousef	MATH102	0
Saleh	Alhamad	STAT101	2
Mishal	Alyousef	STAT101	3

كما أنه ليس من الضروري أن تتمَّ عملية الربط وفق عملية التساوي فحسب، ولكن يمكن إجراء عملية الربـط وفق أيٍّ من عوامل المقارنـة التالية: ؛ إضافةً إلى عامل التساوي. كما يُوجَد أنواع مختلفة من عمليات الربط من ضِمنها الربط الطبيعي، الذي يلغي أحد الحقول المتكرِّرة من النتيجة عند تساوي مُسمَّيات حقول الربط، والربط الخارجي، الذي يكون من ضِمن نتائجه تلك السجلات الموجودة في أحد الجدولين والتي لا يُوجَد ما يقابلها في الجدول الآخر. وسيتم التطرُّق للأنواع المختلفة من عمليات الربط في أثناء شرح لغة الاستفسار البـنائية (في الفصل الثامن).

4-2-2-2-3 عملية القسمة (**Division Operation**):

تُستخدَم عملية القسمة في بعض الحالات الخاصة من الاستفسارات. ويُرمَز لعملية القسمة بالرمز (÷). فعلى سبيل المثال: إذا افترضنا وجود جدولين هما (R) و(S)، وأردنا معرفة الجدول (T) الذي يمثل (R÷S) فإن النتيجة ستكون، كما يلي:

R

X	Y
x1	y1
x2	y1
x3	y1
x4	y1
x1	y2
x3	y2
x2	y3
x3	y3
x4	y3
x1	y4
x2	y4
x3	y4

÷

S

X
x1
x2
x3

⇒

T = R ÷ S

T
y1
y4

وتعني النتيجة السابقة أن عملية القسمة تكافئ العمليات الجبرية الثلاث التالية:

$$T_1 \leftarrow \pi_Y(R)$$
$$T_2 \leftarrow \pi_Y((S \times T_1) - R)$$
$$T \leftarrow T_1 - T_2$$

وبمعنى آخر؛ فإن ناتج عملية القسمة هو جدول (T) يتكوَّن من مجموعة من السجلات؛ بحيث يكون كلُّ سجل في الجدول مدمجاً مع كافة سجلات الجدول (S)، الذي يمثل مقام عملية القسمة، من ضمن سجلات الجدول (R). كما تجدر الملاحظة بأن حقول الجدول (S)، ولتكن A = {a1, a2, …, an}، يجب أن تكون مجموعةً جزئيةً من حقول الجدول (R)، ولتكن B={b1, b2, …, bm}، أيْ: (B).

وعلى افتراض وجود الجدول «موظف» والجدول «مشروع» التاليين، وأننا نرغب في معرفة الموظفين الذين يعملون على كافة المشاريع؛ فإنه يمكن إجراء عملية قسمة للحصول على النتيجة المطلوبة، كما يلي:

EMP

ENO	PNO	PName	Budget
E1	P1	Instrumentation	150000
E2	P1	Instrumentation	150000
E2	P2	Database Develop.	135000
E3	P1	Instrumentation	150000
E3	P4	Maintenance	310000
E4	P2	Instrumentation	150000
E5	P2	Instrumentation	150000
E6	P4	Maintenance	310000
E7	P3	CAD/CAM	250000
E8	P3	CAD/CAM	250000
E3	P2	Database Develop.	135000
E3	P3	CAD/CAM	250000

÷

PROJ

PNO	PName	Budget
P1	Instrumentation	150000
P2	Database Develop.	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

⇒

EMP ÷ PROJ

ENO
E3

وتعني النتيجة السابقة أنه عندما يتمُّ دمج رقم الموظف (E3)، الذي يمثل نتيجة عملية القسمة، مع كافة سجلات المشاريع؛ فإن ناتج عملية الدَّمج هذه ستكون سجلات من ضِمن سجلات جدول الموظفين؛ مما يعني أن الموظف ذا الرقم (E3) يعمل في كافة المشاريع.

ومن الأمثلة الأخرى؛ لنفترض أننا نرغب في معرفة أسماء أعضاء هيئة التدريس المؤهلين لتدريس كافة المواد الدراسية المؤهل لتدريسها عضو هيئة التدريس «محمود السالم» (Mahmood Alsalem) (بما فيهم الموظف «محمود السالم» نفسه). في هذه الحالة يتم أولاً الحصول على كافة بيانات الموظف «محمود السالم» (باستخدام عملية اختيار) ووضعها في الجدول المؤقت (1T) (بما فيها رقم الموظف الذي هو الوسيلة الوحيدة لإجراء عملية الرَّبط مع جدول المؤهلات التدريسية). بعد ذلك؛ يتم إجراء عملية ربط تساوي مع جدول المؤهلات التدريسية متبوعةً بـعملية إسقاط على حقل «رقم المادة الدراسية»؛ وذلك للحصول على أرقام كافة المواد الدراسية المؤهل لتدريسها عضو هيئة التدريس «محمود السالم»، كما يلي:

وتكون نتيجة العمليتين السابقتين، كما يلي:

$$T_1 \longleftarrow \sigma_{FName = \text{"Mahmood"} \text{ AND } LName=\text{"Alsalem"}} (FACULTY_T)$$
$$T_2 \longleftarrow \pi_{Course_ID} (QUALIFICATION_T \bowtie T_1)$$

وللتعرُّف على المؤهلات التدريسية لكافة أعضاء هيئة التدريس؛ تتمُّ عملية إسقاط على حقلي «رقم عضو هيئة التدريس» و«رقم المادة الدراسية» في جدول المؤهلات التدريسية، كما يلي:

T3

FACULTY_	COURSE_
400	CHEM101
420	CHEM102
310	CS101
320	CS102
320	CS103
330	CS104
340	CS105
800	EE101
810	EE102
850	EE103
810	EE104
500	ENGL101
540	ENGL102
560	ENGL103
200	MATH101
200	MATH102
220	MATH103
220	MATH104
220	MATH106
200	MATH107
710	PHYS101
770	PHYS101
730	PHYS102
600	STAT101
660	STAT101
640	STAT102

وتكون نتيجة العملية السابقة، كما يلي:

$$T_4 \longleftarrow T_3 \div T_2$$
$$Result \longleftarrow \pi_{FName,\ LName} (T_4 \bowtie FACULTY_T)$$

بعد ذلك يتمُّ إجراء عملية قسمة بين الجدول المؤقت (3T) والجدول المؤقت (2T)؛ للحصول على أرقام أعضاء هيئة التدريس المؤهلين لتدريس نفس المواد الدراسية المؤهل لتدريسها عضو هيئة التدريس «محمود السالم». وللتعرُّف على أسماء أعضاء هيئة التدريس هؤلاء يتمُّ إجراء عملية رَبط طبيعي بين ناتج عملية القسمة (وهو أرقام أعضاء هيئة التدريس المؤهلين لتدريس نفس المواد التي يقوم بتدريسها عضو هيئة التدريس «محمود السالم») بـجدول أعضاء هيئة التدريس. بعد ذلك

تتمُّ عملية إسقاط على حقلي الاسم الأول واسم العائلة؛ للحصول على جدول النتيجة (Result)، كما يلي:

T_4

```
FACULTY_
--------
710
770
```

وتكون نتيجة العملية الأولى، كما يلي:

Result

FNAME	LNAME
Mahmood	Alsalem
Sultan	Aljasir

أما النتيجة النهائية لما هو مطلوب؛ فتكون، كما يلي:

3–4 الحساب العلاقي (Relational CalCulus):

يستعرض هذا الجزءُ الحساب العلاقي الذي يُعَدُّ اللغة الرسمية الثانية للنموذج العلاقي. وعلى الرغم من وجود نوعين من الحساب العلاقي، وهما: الحساب العلاقي المتعلق بالسجلات (Tuple Relational CalCulus)، والحساب العلاقي المتعلق بالمجال (Domain Relational CalCulus)؛ غير أننا سنستعرض النوع الأول فقط من الحساب العلاقي؛ وذلك لأن اللغة السائدة في التعامل مع بيانات قواعد البيانات العلاقية، وهي لغة الاستفسار البنائية (SQL)، مبنية على الحساب العلاقي المتعلق بالسجلات. وقد تمَّ تطوير كلا النوعين من الحساب العلاقي، بشكلٍ متزامن تقريباً، في مركزين من مراكز أبحاث شركة آي بي أم. أما بالنسبة للحساب العلاقي المتعلق بالمجال؛ فقد تم تطويره للغة أخرى تتعامل مع قواعد البيانات العلاقية وهي لغة «الاستفسار بالمثال» (Query-By-Example (QBE)) التي من أمثلتها تلك المُستخدَمة في نظام إدارة قاعدة بيانات أكسس (ACCESS) المصنَّعة من قِبل شركة ميكروسوفت للحاسبات الشخصية.

إن الحساب العلاقي، وعلى النقيض من الجبر العلاقي الذي سبق شرحه في الجزء السابق، لغة وصفية (DeClarative Language) تُستخدَم لتحديد استفسار ما دون توصيف للطريقة الواجب اتباعها؛ للحصول على نتيجة عملية الاستفسار. ويعني هذا أن أيَّ تعبير حسابي علاقي يحَدِّد ماهية البيانات الواجب استرجاعها عوضاً عن كيفية الوصول إلى البيانات. لذا؛ فإن الحساب العلاقي يُعَدُّ لغةً غير إجرائية (NonproCedural Language). وعلى خلاف ذلك؛ فإن الجبر العلاقي يُعَدُّ لغةً إجرائيةً (ProCedural Language)؛ لأنه يجب علينا كتابة سلسلة من العمليات التي يلزم تنفيذها؛ للوصول إلى البيانات المطلوبة، وأن سلسلة العمليات هذه تحَدِّد ترتيباً للعمليات الموجودة في السلسلة. أمَّا في الحساب العلاقي؛ فإنه يمكِن صياغة الاستفسار بأكثرِ من طريقةٍ، ومع ذلك؛ فإن صياغة الاستفسار لا تؤثر على كيفية الوصول للبيانات المطلوبة. ولذلك؛ فإن اللغة المُستخدَمة للتعامل مع قواعد البيانات، وهي لغة الاستفسار البنائية (SQL)، مبنيةً على أساس هذا النوع من اللغات الرسمية للنموذج العلاقي؛ لكونه يُعفِي المستفيدين من الخوض في تفاصيل كيفية الوصول للبيانات التي يرغبون في الحصول عليها كما يترك المجال مفتوحاً للشركات المطورة لنظم إدارة قواعد البيانات؛ لتطوير طرقٍ مختلفة لتنفيذ استفسارات المستفيدين بشكلٍ فعال. وتُعَدُّ سرعة الاستجابة للاستفسارات المختلفة من قِبل نظم إدارة قواعد البيانات أحد المعايير المهمة التي تميِّز بين نظامٍ وآخر؛ وذلك نتيجةً للاختلافات في فاعلية الطرق المُستخدَمة من قِبل هذه النظم في كيفية الوصول إلى البيانات المطلوبة من قِبل استفسارات المستفيدين.

1-3-4 متغيرات السِّجلات (**Tuple Variables**):

يعتمدُ الحساب العلاقي على ما يُعرَف بمتغيرات السجلات؛ بحيث إن كلَّ «متغير» (Variable) يأخذ قيم سجلات جدول ما، الواحد تلو الآخر؛ بمعنى أن أيَّ متغيرٍ يتمُّ تعريفه يجوب كلَّ سجلات جدول واحد من جداول قاعدة البيانات. فعلى سبيل المثال: الصيغة التالية تمثل استفساراً مكتوباً بالحساب العلاقي:

$$\{t \mid t \in \text{FACULTY_T}\}$$

وتعني الصيغةُ أعلاه أن المتغير (t) قد تمَّ تعريفه؛ بحيث يجوب جدول أعضاء هيئة التدريس (وهو المقصود في الجانب الأيمن من الصيغة)، وستكون نتيجة الاستفسار (وهي الجانب الأيسر من الصيغة) كافة سجلات جدول أعضاء هيئة التدريس. وتُقرأ الصيغة السابقة، كما يلي: ما هي

السجلات (t)؛ بحيث إن (التي يرمز لها بالرمز (|)) هذه السجلات التي سيجوبها المتغير (t) «تنتمي» (∈) لجدول أعضاء هيئة التدريس. ويمكن أن تُوضَع بعض الشروط على السجلات الواجب استرجاعها من جدول أعضاء هيئة التدريس. فعلى سبيل المثال: للحصول على بيانات أعضاء هيئة التدريس الذين يعملون في قسم الحاسب الآلي فقط؛ يمكن صياغة الاستفسار، كما يلي:

$$\{t \mid t \in \text{FACULTY_T} \wedge t[\text{Department_ID}] = \text{“CS”}\}$$

لقد تمَّ تعريفُ المتغير (t) في الصيغة السابقة؛ بحيث يجوب كافة سجلات جدول أعضاء هيئة التدريس، وعندما يكون حقل «رقم القسم الدراسي» للسجل الذي قد أخذ قيمته المتغير مساوياً لقسم الحاسب الآلي؛ يكون هذا السجل من ضِمن سجلات نتيجة الاستفسار. ويمكن النظر للمتغير على أنه مؤشر يشيرُ لسجلات الجدول الواحد تلو الآخر، وعندما تتحقق الشروط الواردة في صيغة الاستفسار، وهي أن يكون القسمُ الذي يتبعه عضو هيئة التدريس هو قسم الحاسب الآلي في مثالنا، يكون السجل من ضِمن سجلات نتيجة الاستفسار. إن الصيغة السابقة للاستفسار تسترجع كافة حقول السجلات التي ينطبق عليها شرط الاسترجاع؛ غير أنه بالإمكان استرجاع بعض حقول السجلات عوضاً عن جميع الحقول. فعلى سبيل المثال: لمعرفة الاسم الأول واسم العائلة لكلِّ عضو هيئة تدريس يعمل في قسم الحاسب الآلي، يُمكِن كتابة الاستفسار وفق الصيغة التالية:

$$\{t[\text{FName}], t[\text{LName}] \mid t \in \text{FACULTY_T} \wedge t[\text{Department_ID}] = \text{“CS”}\}$$

وبشكلٍ عام نحتاج إلى تحديد المعلومات التالية في الحساب العلاقي:

1- لكلِّ متغير نحتاج إلى تحديد الجدول (R) الذي سيجوبه المتغير وفق الصيغة: t R.

2- تحديد شرط الاسترجاع؛ فعندما يجوب المتغير سجلات الجدول الذي تمَّ تعريف المتغير عليه؛ يتمُّ فحص شرط الاسترجاع؛ فإذا كانت النتيجة صحيحة (True)، بمعنى انطباق الشرط على السجل قيد الفحص، يتمُّ إدراج السجل ضِمن نتيجة الاستفسار.

3- تحديد الحقول الواجب إظهارها ضِمن نتيجة الاستفسار؛ بحيث يتمُّ إدراج هذه الحقول ضمن نتيجة الاستفسار لكلِّ سجل ينطبق عليه شرط الاستفسار.

فعلى سبيل المثال: للحصول على رقم هاتف وتاريخ ميلاد كلِّ عضو هيئة تدريس يعمل في الجامعة الأهلية واسم عائلته هو «الصالح» (Alsaleh)، وأن القسم الذي يعمل فيه ليس قسم الحاسب الآلي؛ يمكن كتابة الاستفسار، كما يلي:

$$\{t[\text{Phone_No}], t[\text{DOB}] \mid t \in \text{FACULTY_T} \wedge t[\text{LName}] = \text{“Alsaleh”} \wedge t[\text{Department_ID}] \neq \text{“CS”}\}$$

وتعني الصيغة السابقة أننا نقوم بتحديد الحقول التي ستنتج عن عملية الاستفسار وهي «رقم الهاتف»، و«تاريخ الميلاد» لكل سجل (t) يتم اختياره. بعد ذلك نقوم بتحديد العلاقة التي سيجوبها المتغير والشروط الواجب أن تتحقق في السِّجل (بعد الرمز (|))، وهي في مثالنا الجدول (FaCulty_T) وأن يكون اسم العائلة «الصالح» (Alsaleh)، والقسم الذي يعمل فيه ليس قسم الحاسب الآلي.

2-3-4 التعابير والصيغ في الحساب العلاقي (Expression and Formulae in Relational CalCulus):

إن الشكل العام للتعابير في الحساب العلاقي هو، كما يلي:

$$\{t \mid \mathrm{F}(t)\}$$

بحيث إن الجانب الأيسر للرمز (|) يمثل متغيراً، والجانب الأيمن من الرمز عبارة عن صيغة (Formula) يمكن أن تكون نتيجتها متحققة (True) أو غير متحققة (False). كما يمكن أن يُقيَّد المتغير بحقل أو أكثر مثل: $t[A1]$ و $t[A2]$؛ بحيث إن A1 و A2 تمثلان حقلين من حقول السجلات في الجدول الذي يجوبه المتغير. وعندما تتحقق الصيغة على السجل الذي قد أخذ المتغير قيمته؛ تكون الحقول التي قُيِّدت بـالمتغير من ضمن نتيجة الاستفسار. وتتكون الصيغة من وحدات أو ذرات (Atoms) وعوامل حسابية ومنطقية. وتكون الوحدات، كما يلي:

متغيرات:

- عندما يكون الجدول الذي يجوبه المتغير معروفاً، من الممكن أن يُقيَّد المتغير باسم الجدول، وليكن (R)، كما يلي: $\mathrm{R}.t$ أو $\mathrm{R}(t)$ أو $t \in \mathrm{R}$. (وسنستخدم الصيغة الأخيرة في هذا الكتاب لتمثيل الوحدات التي تُعرف المتغيرات).

شروط:

- عندما يتواجد متغيران، وليكونا s و t، يمكن أن يُربط بينهما بأحد عوامل المقارنة الحسابية θ، وهي: $\{=, <, \leq, >, \geq, \neq\}$، على الشكل التالي: $t[A]\ \theta\ s[B]$.

- عوضاً عن رَبط متغير ما بمتغير آخر باستخدام عوامل المقارنة الحسابية السابقة؛ فإنه يمكن أن يربط المتغير، وليكن t، بقيمة ثابتة (Constant)، ولتكن C، كما يلي: $t[A]\ \theta\ c$.

وباستخدام الوحدات حسب تعريفها أعلاه؛ يمكن أن تؤلف الصِّيغ (Formulae)؛ بحيث تتكون كل صيغة مما يلي:

- وحدات.

- عوامل منطقية: $\wedge, \vee, \neg$.

- عامل الوجود (Existential Quantifier): $\exists$.

- عامل الكل (Universal Quantifier): $\forall$.

أما قواعد تكوين الصيغ؛ فهي، كما يلي:

- كل وحدة تمثل صيغة.

- إذا كانت F و G صيغتين فإن $(F \wedge G)$ و $(F \vee G)$ و$(\neg F)$ و$(\neg G)$ صيغ أيضاً.

- إذا كانت F صيغة فكذلك (F).

- إذا كانت F صيغة وكان t متغيراً في الصيغة فإن $\exists t(F)$ صيغة أيضاً تكون نتيجتها صحيحة (True) إذا وُجد سجل واحد على الأقل في سجلات الجدول الذي تمَّ تعريف المتغير عليه؛ بحيث تنطبق عليه الصيغة F. وخلاف ذلك تكون النتيجة خطأ (False). كما يمكن كتابة الصيغة $\exists t(F)$ على الشكل (t) $\exists t(F)$.

- إذا كانت F صيغة وكان t متغيراً في الصيغة؛ فإن $\forall t(F)$ تُعَدُّ صيغةً أيضاً تكون نتيجتها صحيحة (True) إذا انطبقت الصيغة F على كافة سجلات الجدول الذي تمَّ تعريف المتغير عليه. وخلاف ذلك تكون النتيجة خطأ (False). كما يمكن كتابة الصيغة $\forall t(F)$ على الشكل $\forall t\ F(t)$.

4-3-2-1 التعابير الآمنة (Safe Expressions):

عندما يستخدم عامل الوجود، وعامل الكل، ونفي الشروط في تعبيرات الحساب العلاقي؛ فإنه من الضروري التأكُّد من أن التعبيرات الحسابية ذات معنى. والتعبير الآمن في الحساب العلاقي يضمن أن تكون نتيجته عدداً محدداً من السجلات. وخلاف ذلك؛ فإن التعبير الحسابي يُعَدُّ غير آمن. فعلى سبيل المثال: يُعَدُّ التعبير التالي غير آمن؛ لكون نتيجته عدد غير مُحدَّد من السجلات:

$$\{t \mid \neg t \in R\}$$

وتكون نتيجة التعبير السابق هي كافة السجلات التي من الممكن أن تُوجَد؛ ولكنها ليست من ضمن السجلات الموجودة فعلياً في الجدول (R). وهذه السجلات بالطبع ذات عدد غير محدود. ويمكن تعريف التعبير الآمن بشكل أكثر تحديداً، كما يلي:

يُعَدُّ التعبيرُ الحسابي آمناً إذا كان بالإمكان حساب نتيجته باستخدام قيم ثابتة فقط من ضِمن القيم الموجودة في قاعدة البيانات، أو من ضِمن التعبير الحسابي نفسه.

ولكون قاعدة البيانات تحتوي على عددٍ محددٍ من القيم؛ فإن القيم الثابتة المخزَّنة فيها محدودة. وكذلك هو الحال بالنسبة لعدد القيم الثابتة التي من الممكن أن يحتويها التعبير الحسابي التي لا بد أن تكون محدودةً أيضاً. ونتيجة لذلك؛ فإن التعبير الحسابي سيكون آمناً من حيث إن نتيجته ستحتوي على عددٍ مُحدَّدٍ من القيم إذا احتوى على قيم ثابتة من قيم قاعدة البيانات أو قيم ثابتة ضمنه.

ولنفترض وجود الجداول الأربعة التالية ضمن قاعدة البيانات التي تمثل جدول الموظفين، وجدول المشاريع، وجدول المخصصات المالية، وجدول الأعمال التي يقوم بها كلُّ موظف ضمن كل

مشروع يشارك فيه. ولنفترض كذلك أننا نرغب في إجراء بعض الاستفسارات، مستخدمين الحساب العلاقي.

Employee (Eno, Ename, Title, City)

ProjeCt (Pno, Pname, Budget, City)

Payment (Title, Salary)

Job (Eno, Pno, Responsibility, Duration)

الاستفسار الأول: ما أسماء كافة الموظفين؟

الحل:

$\{t[\text{Ename}] \mid t \in \text{Empoyee}\}$

اقترن المتغير (t) في الجانب الأيسر من التعبير الحسابي، في الحل أعلاه، بحقل «اسم الموظف» (Ename)؛ مما يعني أن سجلات النتيجة النهائية للتعبير الحسابي ستتضمَّن حقل «اسم الموظف» فقط؛ وذلك لكل سجل تنطبق عليه الصيغة المعرفة في الجانب الأيمن من التعبير الحسابي. أمَّا الصيغة التي يجب أن تتحقق على السجل حتى يكون من ضمن سجلات النتيجة النهائية للتعبير الحسابي؛ فلم تتضمَّن أية شروط. لذلك؛ فإنه عندما يجوب المتغير (t) سجلات جدول الموظفين ($t \in$ Empoyee)، الواحد تلو الآخر، سيكون كلُّ سجل من سجلاته مؤهلاً لأن يكون من ضمن نتيجة التعبير الحسابي. وبناءً على ذلك ستحتوي النتيجة النهائية للتعبير الحسابي على سجلات تتضمَّن كافة أسماء الموظفين.

الاستفسار الثاني: ما أسماء المشاريع وميزانياتها؟

الحل:

$\{t[\text{Pname}], t[\text{Budget}] \mid t \in \text{Project}]\}$

اقترن المتغير (t) في الجانب الأيسر من التعبير الحسابي، في الحل أعلاه، بحقل كلٍّ من «اسم المشروع» (Pname) وحقل «ميزانية المشروع» (Budget)؛ مما يعني أن سجلات النتيجة النهائية للتعبير الحسابي ستتضمَّن حقل «اسم المشروع»، وحقل «ميزانية المشروع» فقط؛ وذلك لكلِّ سجل تنطبق عليه الصيغة المعرفة في الجانب الأيمن من التعبير الحسابي. أمَّا الصيغة التي يجب أن تتحقق على السجل حتى يكون من ضمن سجلات النتيجة النهائية للتعبير الحسابي؛ فلم تتضمَّن أية شروط. لذلك؛ فإنه عندما يجوب المتغير (t) سجلات جدول المشاريع ($t \in$ Project)، الواحد تلو الآخر؛ سيكون كلُّ سجل من سجلاته مؤهلاً لأن يكون من ضمن نتيجة التعبير الحسابي. وبناءً على ذلك ستتضمَّن النتيجة النهائية للتعبير الحسابي سجلاتٍ تحتوي على كافة أسماء المشاريع وميزانياتها.

الاستفسار الثالث: ما مُسميات الوظائف التي تمَّ تعيين موظف واحد على الأقل في كل واحدة منها؟

الحل:

$$\{t[\text{Title}] \mid t \in \text{Employee})\}$$

يحتوي جدول الموظفين على سجلات لكافة الموظفين متضمناً ذلك مُسمَّيات الوظائف التي تم تعيينهم عليها. ويعني هذا أنَّ كلَّ مُسمَّى وظيفي مُدرَج في جدول الموظفين يعني ضمنياً وجود موظفٍ واحدٍ على الأقل معين على مُسمَّى هذه الوظيفة. ولحل الاستفسار، في هذه الحالة، يُكتفَى بـمعرفة مسميات الوظائف المدرَجة في جدول الموظفين. ولمعرفة مُسميات الوظائف هذه؛ اقترن المتغير (t) في الجانب الأيسر من التعبير الحسابي بحقل «مسمى الوظيفة» (Title)، في الحل أعلاه؛ مما يعني أن سجلات النتيجة النهائية للتعبير الحسابي ستتضمَّن حقل «مُسمَّى الوظيفة» فقط؛ وذلك لكلِّ سجل تنطبق عليه الصيغة المعرفة في الجانب الأيمن من التعبير الحسابي. أمَّا الصيغة التي يجب أن تتحقق على السجل حتى يكون من ضِمن سجلات النتيجة النهائية للتعبير الحسابي؛ فلم تتضمَّن أي شروط. لذلك؛ فإنه عندما يجوب المتغير (t) سجلات جدول الموظفين ($t \in$Employee)، الواحد تلو الآخر؛ سيكون كل سجل من سجلاته مؤهلاً لأن يكون من ضمن نتيجة التعبير

الحسابي. وبناءً على ذلك؛ ستتضمَّن النتيجة النهائية للتعبير الحسابي سجلاتٍ تحتوي على كافة مسميات الوظائف للموظفين ذوي السجلات في جدول الموظفين. وهذا يعني أن النتيجة ستحتوي على مُسمَّيات كافة الوظائف التي تم تعيين موظف واحد على الأقل على كل منها.

الاستفسار الرابع: ما بيانات الموظفين الذين يقطنون في مدينة الدمام؟

الحل:

$$\{t \mid t \in \text{Employee} \land t[\text{City}] = \text{“Dammam”}\}$$

لم يقترن المتغير (t) في الجانب الأيسر من التعبير الحسابي، في الحل أعلاه، بأيِّ حقلٍ من حقول سجلات الجدول الذي يجوبه المتغير. لذلك؛ فإن النتيجة النهائية للتعبير الحسابي ستتضمَّن كافة حقول سجلات الجدول التي تنطبق عليها الصيغة المعرفة في الجانب الأيمن من التعبير الحسابي. أما الصيغة التي يجب أن تتحقق على السجل حتى يكون من ضِمن سجلات النتيجة النهائية للتعبير الحسابي؛ فتضمَّنت شرط كون قيمة حقل «المدينة» (City) مساوية لمدينة «الدمام» (Dammam). لذلك؛ فإنه عندما يجوب المتغير (t) سجلات جدول الموظفين ($t \in \text{Employee}$)، الواحد تلو الآخر؛ سيكون كلُّ سجل قيمة حقل المدينة فيه مساوية لمدينة الدمام مؤهلاً لأن يكون من ضِمن نتيجة التعبير الحسابي. وبناءً على ذلك ستتضمَّن النتيجة النهائية للتعبير الحسابي كافة بيانات الموظفين الذين يقطنون في مدينة الدمام.

الاستفسار الخامس: ما المدن التي يقطنها موظفون، ويتوفر فيها مشاريع؟

الحل:

$$\{t[\text{City}] \mid t \in \text{Employee} \land \exists s(s \in \text{Project} \land t[\text{City}] = s[\text{City}])\}$$

أو

$$\{t[\text{City}] \mid t \in \text{Project} \land \exists s(s \in \text{Employee} \land t[\text{City}] = s[\text{City}])\}$$

لحلِّ هذا الاستفسار نحتاجُ إلى تعريف متغيرين: الأول منهما يجوب جدول الموظفين، والثاني يجوب جدول المشاريع. وعندما يكون حقل المدينة في السجل الذي أخذ قيمته المتغير الذي يجوب

جدول الموظفين مساوياً لحقل المدينة في أحد سجلات المتغير الذي يجوب جدول المشاريع؛ فإن هذا يعني وجود موظف ومشروع في نفس المدينة، وتكون المدينة من ضِمن نتيجة التعبير الحسابي.

اقترن المتغير (t) في الجانب الأيسر من التعبير الحسابي، في الحل الأول أعلاه، بحقل «المدينة» (City)؛ مما يعني أن سجلات النتيجة النهائية للتعبير الحسابي ستتضمَّن حقل «المدينة» فقط؛ وذلك لكلِّ سجل تنطبق عليه الصيغة المعرَّفة في الجانب الأيمن من التعبير الحسابي. أما الصيغة التي يجب أن تتحقق على السجل حتى يكون من ضِمن سجلات النتيجة النهائية للتعبير الحسابي؛ فشرحها كما يلي:

- يجوب المتغير (t) جدول الموظفين السجل تلو الآخر. ويعني هذا أن حقل «المدينة» في النتيجة النهائية هو حقل «المدينة» التابع لجدول الموظفين؛ لكون الحقل مرتبطاً بالمتغير (t) الذي تم تعريفه؛ بحيث يجوب سجلات جدول الموظفين.

- لكل سجل يأخذ قيمته المتغير (t) يجب أن يتحقق الشرط التالي:

• يجوب المتغير (s) جدول المشاريع، وعندما يوجد (∃) سجل في جدول المشاريع تكون المدينة الموجودة فيه مساويةً لمدينة الموظف في السجل الذي توقف عنده المتغير (t)؛ يكون سجل الموظف من ضمن سجلات جدول النتيجة النهائية التي يُؤخَذ منها قيمة حقل المدينة فقط.

- تستمر العملية حتى ينتهي المتغير (t) من المرور بكافة سجلات الموظفين.

الحل الثاني للاستفسار مماثلٌ للحلِّ الأول، ومكافئ لنتيجته؛ غير أنه يُلاحَظ في الحل الثاني عكس ترتيب الجداول التي يجوبها المتغيران؛ فالمتغير (t) يجوب جدول المشاريع عوضاً عن جدول الموظفين، والمتغير (s) يجوب جدول الموظفين عوضاً عن جدول المشاريع. أما حقل المدينة، الذي يمثل نتيجة التعبير الحسابي؛ فهو حقل جدول المشاريع عوضاً عن حقل جدول الموظفين.

الاستفسار السادس: ما المدنُ التي يُوجَد فيها مشاريع ولا يقطنها أيٌّ من الموظفين؟

الحل:

$$\{t[\text{City}] \mid t \in \text{Project} \wedge \neg\exists s(s \in \text{Employee} \wedge t[\text{City}] = s[\text{City}])\}$$

لحلِّ هذا الاستفسار نحتاج إلى تعريف متغيرين: الأول منهما يجوب جدول المشاريع، والثاني يجوب جدول الموظفين. وعندما لا يُوجَد لقيمة حقل المدينة في السجل الذي أخذ قيمته المتغير الذي يجوب جدول المشاريع ما يساويها من قيمة في حقل المدينة في كافة سجلات المتغير الذي يجوب جدول الموظفين؛ فإن هذا يعني وجود مشروع في المدينة مع عدم وجود ما يقطنها من موظفين.

اقترن المتغير (t) في الجانب الأيسر من التعبير الحسابي، في الحل أعلاه، بحقل «المدينة» (City)؛ مما يعني أن سجلات النتيجة النهائية للتعبير الحسابي ستتضمَّن حقل «المدينة» فقط؛ وذلك لكلِّ سجل تنطبق عليه الصيغة المعرفة في الجانب الأيمن من التعبير الحسابي. أما الصيغة التي يجب أن تتحقق على السجل حتى يكون من ضِمن سجلات النتيجة النهائية للتعبير الحسابي فشرحها، كما يلي:

- يجوب المتغير (t) جدول المشاريع السجل تلو الآخر. ويعني هذا أن حقل «المدينة» في النتيجة النهائية هو حقل «المدينة» التابع لجدول المشاريع؛ لكون الحقل مرتبطاً بالمتغير (t) الذي تمَّ تعريفه؛ بحيث يجوب سجلات جدول المشاريع.

- لكلِّ سجل يأخذ قيمته المتغير (t) يجب ألا يتحقق الشرط التالي (بمعنى أن السجل لن يكون من ضمن النتيجة النهائية للتعبير الحسابي):

• يجوب المتغير (s) جدول الموظفين، وعندما يوجد (∃) سجل في جدول الموظفين تكون المدينة الموجودة فيه مساويةً لمدينة المشروع في السجل الذي توقف عنده المتغير (t)، يكون سجل المشروع من ضِمن سجلات جدول النتيجة النهائية التي يُؤخَذ منها قيمة حقل المدينة فقط.

- تستمرُّ العملية حتى ينتهي المتغير (t) من المرور بكافة سجلات الموظفين.

وتتمُّ عملية استثناء سجلٍّ ما من الدخول في النتيجة النهائية للتعبير الحسابي باستخدام عامل النفي المنطقي (¬)؛ مما يعني أن عامل النفي إذا سبق عامل الوجود تكون نتيجته «عدم الوجود».

ويُلاحَظ في هذا المثال عدم وجود حَلٍّ ثانٍ مماثلٍ لحل المثال الخامس؛ لأننا لو عكسنا ترتيب الجداول المرتبطة بالمتغيرات؛ فستكون النتيجة مكافئة للاستفسار التالي:

الاستفسار: ما المدن التي يقطنها موظفون ولا يُوجَد فيها مشاريع؟

ونتيجة الاستفسار السابق مختلفة، بكل تأكيد، عن المطلوب في الاستفسار الأساسي.

الاستفسار السابع: ما أسماء المشاريع التي تزيد ميزانياتها على (250.000)؟

الحل:

$$\{t[\text{Pname}] \mid t \in \text{Project} \wedge t[\text{Budget}] > 250000\}$$

اقترن المتغير (t) في الجانب الأيسر من التعبير الحسابي، في الحلِّ أعلاه، بحقل «اسم المشروع» (Pname)؛ مما يعني أن سجلات النتيجة النهائية للتعبير الحسابي ستتضمَّن حقل «اسم المشروع» فقط؛ وذلك لكلِّ سجل تنطبق عليه الصيغة المعرفة في الجانب الأيمن من التعبير الحسابي. أمَّا الصيغة التي يجب أن تتحقق على السجل حتى يكون من ضمن سجلات النتيجة النهائية للتعبير الحسابي؛ فتضمَّنت شرط كون قيمة حقل «ميزانية المشروع» (Budget) أكبر من (250.000). لذلك؛ فإنه عندما يجوب المتغير (t) سجلات جدول المشاريع ($t \in$ ProjeCt)، الواحد تلو الآخر، سيكون كل سجل قيمة حقل ميزانيته أكبر من (250.000) مؤهلاً لأن يكون من ضمن نتيجة التعبير الحسابي. وبناءً على ذلك ستتضمَّن النتيجة النهائية للتعبير الحسابي أسماء كافة المشاريع التي تزيد ميزانياتها على (250.000).

الاستفسار الثامن: ما أسماء وميزانيات المشاريع التي يعمل فيها الموظف رقم (E1)؟

الحل:

$$\{t[\text{Pname}], t[\text{Budget}] \mid t \in \text{Project} \wedge \exists s(s \in \text{Job} \wedge t[\text{Pno}]=s[\text{Pno}] \wedge s[\text{Eno}]=\text{`E1`})\}$$

اقترن المتغير (t) في الجانب الأيسر من التعبير الحسابي، في الحلِّ أعلاه، بحقل كلٍّ من «اسم المشروع» (Pname) وحقل «ميزانية المشروع» (Budget)؛ مما يعني أن سجلات النتيجة النهائية للتعبير الحسابي ستتضمَّن حقل «اسم المشروع» وحقل «ميزانية المشروع» فقط؛ وذلك لكلِّ سجل تنطبق عليه الصيغة المعرفة في الجانب الأيمن من التعبير الحسابي. أمَّا الصيغة التي يجب أن تتحقق على السجل حتى يكون من ضِمن سجلات النتيجة النهائية للتعبير الحسابي فشرحها، كما يلي:

- يجوب المتغير (t) جدول المشاريع السجل تلو الآخر. ويعني هذا أن كلاً من حقل «اسم المشروع» (Pname)، وحقل «ميزانية المشروع» (Budget) في النتيجة النهائية للتعبير الحسابي هما من حقول جدول المشاريع؛ وذلك لكونهما مرتبطين بالمتغير (t) الذي تم تعريفه؛ بحيث يجوب سجلات جدول المشاريع.

- لكلِّ سجل يأخذ قيمته المتغير (t) يجب أن يتحقق الشرط التالي:

• يجوب المتغير (s) جدول «الأعمال» (Job)، وعندما يوجد (∃) سجل في جدول الأعمال تكون قيمة حقل «رقم المشروع» (Pno) فيه مساويةً لقيمة حقل «رقم المشروع» (Pno) في السجل الذي توقف عنده المتغير (t)، وتكون قيمة حقل «رقم الموظف» (Eno) في السجل الذي أخذ قيمته المتغير (s) هي "E1"، يكون سجل المشروع من ضِمن سجلات جدول النتيجة النهائية التي يُؤخَذ منها قيمة حقل «اسم المشروع» وحقل «ميزانية المشروع».

- تستمر العملية حتى ينتهي المتغير (t) من المرور بكافة سجلات المشاريع.

4-4 أمثلة على استخدام الجبر العلاقي والحساب العلاقي:

يتوفر لإحدى مؤسسات تأجير العقارات عددٌ من المكاتب في مدن مختلفة، وعددٌ من العاملين في المؤسسة. كما تحتفظ المؤسسة ببيانات عن عملائها الذين يتقدَّمون لها بطلبات لاستئجار العقارات التي تُشرف عليها بالإضافة لبيانات العقارات التي تُشرف عليها وبيانات مالكي هذه العقارات. كما أن المؤسسة تحتفظ ببيانات عن المواعيد التي يتمُّ تحديدها للعملاء المختلفين لمعاينة العقارات التي تُشرف عليها قبل قيامهم باستئجار ما يتناسب من العقارات التي تشرف عليها المؤسسة مع احتياجاتهم. وتحتفظ المؤسسة بهذه البيانات ضِمن قاعدة بيانات علاقية تتكون من الجداول التالية (Connolly and Begg, 2015):

1- جدول مكاتب المؤسسة:

BranCh (BranChNo: integer, Street: string, City: string, PostCode: string)

2- جدول العاملين في المؤسسة:

Staff (StaffNo: integer, FName: string, LName: string, Position: string, Sex: string, DOB: date, Salary: integer, BranChNo: integer)

3- جدول العقارات التي تُشرف عليها المؤسسة:

PropertyNo: integer, Street: string, City: string,PropertyForRent (
PostCode: integer, Type: string, NumberOfRooms, integer, Rent: integer, OwnerNo: integer, StaffNO: integer, BranChNo: integer)

4- جدول العملاء (أو المستأجرين):

Client (ClientNo: integer, FName: string, LName: string, TelNo: string, PropertyPreferenCe: string, MaxRent: integer)

5- جدول ملاك العقارات:

Owner (OwnerNo: integer, FName: string, LName: string, Address: string, TelNo: string)

6- جدول مواعيد معاينة العقارات من قِبل العملاء (تدخل المواعيد بعد الانتهاء من المعاينة():

ViewingSChedule (ClientNo: integer, PropertyNo: integer, ViewDate: date, Comments: string)

وبناءً على الجداول السابقة لقاعدة بيانات المؤسسة، المطلوب هو الإجابة عن الاستفسارات التالية باستخدام (1) الجبر العلاقي، و(2) الحساب العلاقي.

الاستفسار الأول: ما الأسماء الأولى وأسماء عائلات العاملين الذين يشغلون مناصب إشرافية (Manager) في المؤسسة ويتقاضون رواتب تزيد على (50.000)؟

الحل:

$1- \pi_{FName, LName}(\sigma_{Position = \text{"Manager"} \wedge Salary > 50000}(Staff))$

$2- \{t[FName], t[LName] \mid t \subset Staff \wedge t[Position] = \text{"Manager"} \wedge t[Salary] > 50000\}$

الاستفسار الثاني: ما الأسماء الأولى وأسماء عائلات العاملين الذين يشرفون على عقارات معروضة للإيجار في مدينة الرياض؟

الحل:

$1- \pi_{FName, LName}(\sigma_{Staff.StaffNo = PropertyForRent.StaffNo}(Staff \times \sigma_{City = \text{"Riyadh"}}(PropertyForRent)))$

ملاحظة: يمكن استخدام إعادة تسمية حقل «رقم الموظف» في أحد الجدولين عوضاً عن استخدام اسم الجدول متبوعاً باسم الحقل لحل هذا المثال، كما يلي:

$1- \pi_{FName, LName}(\sigma_{StaffNo = Staff_ID}(Staff \times \rho_{StaffNo = Staff_ID}(\sigma_{City = \text{"Riyadh"}}(PropertyForRent))))$

$2- \{t[FName], t[LName] \mid t \in Staff \wedge \exists s (s \in PropertyForRent \wedge s[StaffNo] = t[StaffNo] \wedge s[City] = \text{"Riyadh"})\}$

الاستفسار الثالث: ما الأسماء الأولى وأسماء عائلات العاملين في المؤسسة الذين لا يشرفون حالياً على أية عقارات معروضة للإيجار؟

الحل:

$1- \pi_{FName, LName}(Staff - (Staff \bowtie \pi_{StaffNo}(PropertyForRent)))$

$2- \{t[FName], t[LName] \mid t \in Staff \wedge \neg \exists s (s \in PropertyForRent \wedge s[StaffNo] = t[StaffNo])\}$

الاستفسار الرابع: ما الأسماء الأولى وأسماء عائلات العملاء الذين قاموا بمعاينة عقارات في مدينة الرياض؟

الحل:

$1- \pi_{FName, LName}(Client \bowtie (ViewingSchedule \bowtie \sigma_{City = \text{"Riyadh"}}(PropertyForRent)))$

$2- \{t[FName], t[LName] \mid t \in Client \wedge \exists s \exists u (s \in ViewingSchedule\ u \in PropertyForRent \wedge s[ClientNo] = t[ClientNo] \wedge s[PropertyNo] = u[PropertyNo] \wedge u[City] = \text{"Riyadh"})\}$

الاستفسار الخامس: ما المدن التي يتوفر فيها مكتب للمؤسسة وعقار واحد على الأقل معروض للإيجار فيها؟

الحل:

1- $\pi_{City}(Branch) \cap \pi_{City}(PropertyForRent)$

2- $\{t[City] \mid t \in Branch \land \exists s\,(s \in PropertyForRent \land s[City] = t[City])\}$

الفصل الخامس

التصميم المنطقي لنظم قواعد البيانات العلاقية

تتكوَّن مرحلة التصميم المنطقي لقواعد البيانات من خطوتين رئيسيتين. في الخطوة الأولى يتمُّ تحويل النموذج المفاهيمي إلى نموذج قاعدة البيانات المُستخدَمة. ولأن هذا الكتاب يركِّز على قواعد البيانات العلاقية؛ فإن هذه الخطوة تعني تحويل النموذج المفاهيمي إلى النموذج العلاقي. أمَّا في الخطوة الثانية؛ فيتمُّ تحسين تصميم قاعدة البيانات الناتجة من عملية التحويل؛ بحيث تحتوي على أقلِّ قدرٍ ممكن من البيانات المتكرِّرة؛ حتى يتمَّ تجنُّب الأخطاء التي قد تنتج عن عمليات التعديل على محتويات قاعدة البيانات. وتُدعَى هذه الخطوة بعملية «التطبيع» (Normalization).

ويركِّز هذا الفصل من الكتاب على الخطوة الأولى من عملية التصميم المنطقي لقواعد البيانات؛ بحيث يتمُّ تصميم هياكل قاعدة البيانات العلاقية بناءً على تصميمها المُسبق باستخدام النموذج المفاهيمي. ويُطلَق على هذه الخطوة في بعض الأحيان عملية «التحويل بين النماذج». ولا تقتصر هذه الخطوة على قواعد البيانات العلاقية؛ ولكن يمكن أن تُطبق على أيِّ نماذج أخرى لقواعد البيانات التمثيلية، مثل: الشبكية، والهرمية، والشيئية. فهذه الخطوة من مرحلة التصميم المنطقي ما هي إلا عملية تصميم لهياكل قاعدة البيانات وفق النموذج التمثيلي المُستخدَم. ولأن النموذج العلاقي هو أحد المحاور الرئيسية لهذا الكتاب؛ فإن هذا الفصل يركِّز على عملية التصميم المنطقي لقواعد البيانات العلاقية. ولكون عملية تصميم نظم قواعد البيانات تأتي عادةً بشكلٍ متسلسلٍ؛ فإن هذا الجزء يركز على عملية تحويل النموذج المفاهيمي، القريب من مستوى إدراك وفهم المستفيدين لقاعدة البيانات التي تمثل البيانات التي يتعاملون معها والقيود المفروضة عليها،

إلى النموذج العلاقي، الذي يتم التعامل معه من قِبل المتخصصين في نظم قواعد البيانات والمستفيدين ذوي الخبرة في مجال الحاسب الآلي.

ويعتمد العديدُ من أدوات هندسة البرمجيات (Computer-Aided Software Engineering (CASE) Tools) على نموذج «كينونة - علاقة»، أو نماذج شبيهة، في عملية التصميم المفاهيمي لقواعد البيانات. وتمكِّن هذه الأدوات مُصمِّمي نظم قواعد البيانات من تصميم قواعد البيانات بشكلٍ تفاعلي (InteraCtive) على هيئة رسومات نموذج «كينونة - علاقة»، التي سبق شرحها في الفصلين الثاني والثالث. كما تمكن هذه الأدوات، وبشكلٍ آلي، من تحويل النموذج المفاهيمي الذي تمَّ تصميمه إلى هياكل قاعدة بيانات علاقية مُستخدِمَةً لغة تعريف البيانات (Data Definition Language)، التي تُعَدُّ جزءاً من لغة الاستفسار البنائية (SQL)، الخاصة بقاعدة البيانات العلاقية المُستخدَمة. ولإجراء عملية التحويل هذه؛ تستخدم أدوات هندسة البرمجيات خطواتٍ شبيهةً بخطوات التحويل التي يتطرق إليها هذا الفصل. أمَّا الخطوة الثانية من مرحلة التصميم المنطقي والمتمثلة في عملية التطبيع؛ فهي محور الجزء الأول من الفصل السادس.

5–1 التحويل من النموذج المفاهيمي «كينونة – علاقة» إلى النموذج العلاقي:

خلال عملية التصميم المنطقي لقاعدة البيانات؛ يتمُّ تحويل النموذج المفاهيمي «كينونة - علاقة» إلى هياكل قاعدة بيانات علاقية. وتكون المدخلات لهذه العملية هي نموذج بيانات «كينونة - علاقة»، أمَّا مُخرجات هذه العملية؛ فهي هياكل قاعدة البيانات. كما أن هذه العملية تُعَدُّ عمليةً بسيطةً إلى حَدٍّ ما، ولها قواعدها المعروفة لدرجة أن الكثير من أدوات هندسة البرمجيات (CASE Tools) تقوم بعملية التحويل هذه بشكلٍ آلي، كما أسلفنا سابقاً. غير أنه من الضروري التعرُّف على خطوات هذه العملية لثلاثة أسباب (Hoffer et al, 2018):

1- لا تستطيع معظم أدوات هندسة البرمجيات نمذجة علاقات معقدة، مثل: العلاقات الثلاثية، وعلاقات الأنواع الرئيسية والأنواع الفرعية؛ ومن ثمَّ تحويلها إلى جداول علاقية. في مثل هذه الحالات قد يَتطلَّب الأمر تحويلَ مثل هذه العلاقات بشكلٍ يدوي.

2- قد يتوفَّر عددٌ من البدائل لتحويل بعض الحالات في النموذج المفاهيمي التي يمكن اختيارُ المناسب منها مع الوضع الذي نحاول نمذجته.

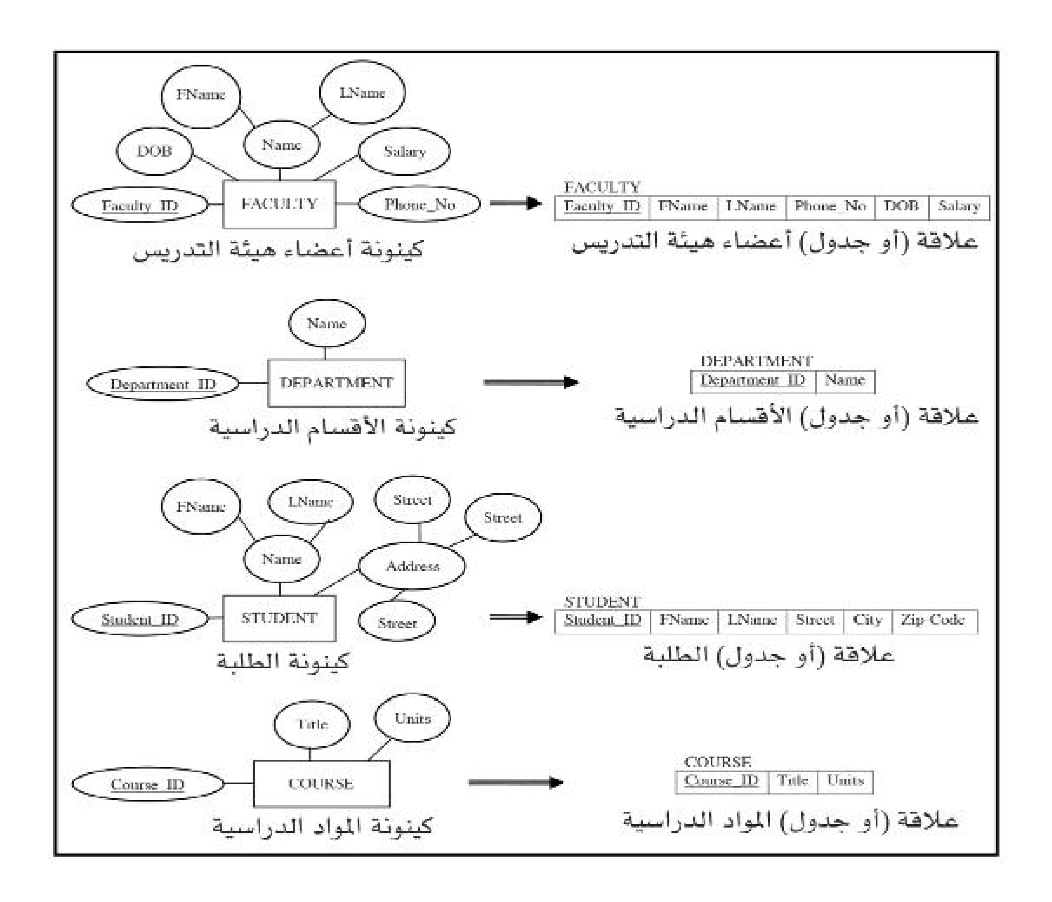
3- قد يتطلَّب الأمرُ التأكُّد من جودة مخرجات أدوات هندسة البرمجيات؛ من حيث إنها قد قامت بـتحويل النموذج المفاهيمي إلى جداول جيدة التصميم.

وفيما يلي سنوضِّح عملية التحويل من النموذج المفاهيمي «كينونة ـ علاقة» إلى النموذج العلاقي؛ مُستخدِمين الأمثلة التي سبق التطرُّق إليها في الفصلين الثاني والثالث، مع التركيز على الحالة الدراسية المتعلقة بالجامعة الأهلية.

5–1–1 قاعدة التحويل الأولى: التعامل مع الكينونات القوية (أو العادية) وخصائصها:

لكلِّ كينونة قوية موجودة ضِمن النموذج المفاهيمي «كينونة ـ علاقة» يتمُّ إنشاء جدول يحمل مُسمَّى الكينونة، ويحتوي على جميع الخصائص البسيطة المرتبطة بالكينونة باعتبارها حقولاً ضِمن الجدول قيد الإنشاء. أمَّا بالنسبة للخصائص المركبة؛ فيتمُّ إنشاء حقول لمكوناتها البسيطة فقط ضِمن الجدول. أمَّا خاصية المعرِّف؛ فتصبح المفتاح الرئيسي للجدول قيد الإنشاء. وإذا وُجِدَت خاصية مشتقة ضِمن خصائص كينونة معينة؛ فإنَّ مثل هذه الخاصية لا يتم إنشاءُ حقل مقابلٍ لها في الجدول قيد الإنشاء؛ وذلك لأنه بالإمكان حساب (أو استخلاص) قيمة هذه الخاصية من خلال الحقول الأخرى التي تمَّ إنشاؤها في الجدول؛ غير أنه يجب الملاحظة أن مثل هذه الخاصية المشتقة يجب تمثيلها وعدم تجاهلها في النموذج المفاهيمي للإشارة إلى أن قيمتها ذات أهمية، وأنه سيتمُّ حسابها من قِبل التطبيقات التي ستتعامل مع قاعدة البيانات مستقبلاً.

ولأنه يُوجَد لدينا في النموذج المفاهيمي للجامعة الأهلية أربعة كينونات قوية، وهي: كينونة «عضو هيئة التدريس» (FACULTY)، وكينونة «القسم الدراسي» (DEPARTMENT)، وكينونة «الطالب» (STUDENT)، وكينونة «المادة الدراسية» (COURSE)؛ فإنه يتمُّ تحويلها إلى أربعة جداول علاقية حسب قواعد تحويل الكينونات القوية أعلاه. والشكل رقم (5-1) يوضِّح عملية تحويل الكينونات الأربع، مع ملاحظة وضع خط متصل تحت مُسمَّى الحقل الذي يمثل المفتاح الرئيسي لكلِّ جدول.



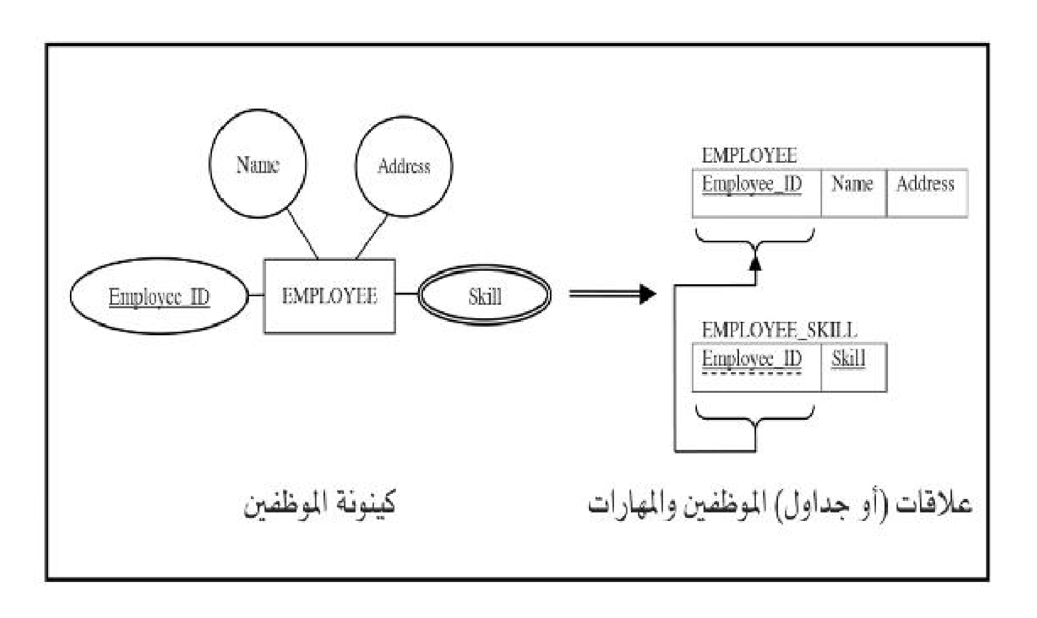
شكل رقم (5–1): تحويل الكينونات القوية إلى علاقات (أو جداول).

5–1–1–1 التعامل مع الخاصية المتعددة القيم:

في حالِ تضمَّنت إحدى الكينونات القوية خاصيةً متعددة القيم؛ فإنه يتمُّ إنشاء جدولين عوضاً عن جدولٍ واحدٍ؛ بحيث يُسمَّى الجدول الأول باسم الكينونة قيد التحويل، وتكون حقوله ممثلةً لكافة خصائص الكينونة ما عدا الخاصية المتعددة القيم. أمَّا الجدول الثاني؛ فيتكون من حقلين مجتمعين يمثلان المفتاح الرئيسي للجدول؛ بحيث يكون أحدهما ممثلاً للمفتاح الرئيسي للجدول الأول، ويكون في الوقت نفسه مفتاحاً خارجياً يشير إلى الجدول الأول. أمَّا الحقل الثاني؛ فيمثل الخاصية المتعددة القيم. ويستمدُّ الجدول الثاني مُسمَّاه من معنى الخاصية المركبة (أو اسمها الفعلي).

ويُوضِّح الشكل رقم (5-2) كينونة «الموظف» (EMPLOYEE) التي تتضمَّن خاصيةً متعددة القيم، وهي خاصية «المهارة» (Skill)؛ للدلالة على أنه قد يكون للموظف الواحد أكثر من

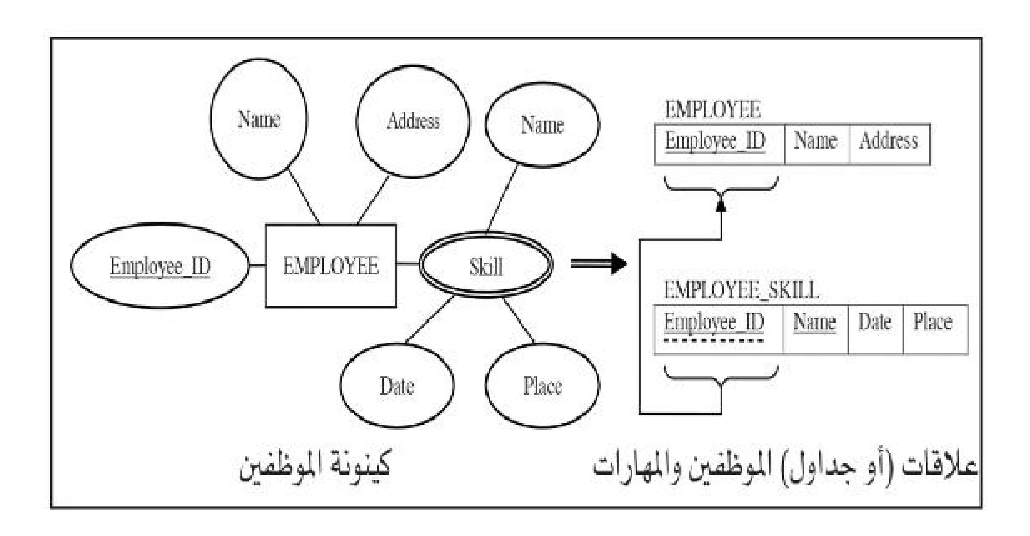
مهارة (مثل: البرمجة بلغة باسكال، وكوبول، وسي... إلخ). وباتباع قواعد التحويل السابقة؛ يتمُّ إنشاء جدولين. الجدول الأول يحملُ مُسمَّى «موظف» (EMPLOYEE) ويحتوي على حقولٍ تمثل جميع خصائص كينونة «الموظف» ما عدا الخاصية المتعددة القيم. أمَّا الجدول الثاني واسمه «مهارة الموظف» (EMPLOYEE_SKILL)؛ فيحتوي على حقلين: الأول منهما هو المفتاح الرئيسي للجدول الأول (EMPLOYEE) وهو (Employee_ID). والثاني فيمثل المهارة ويحمل مُسمَّى «مهارة» (Skill). ويمثل كلا الحقلين مدمجين أحدهما مع الآخر المفتاح الرئيسي للجدول؛ إذ يتمُّ توضيح ذلك من خلال وضع خط متصل تحت كلٍّ منهما. أمَّا الخط المتقطع تحت حقل «رقم الموظف» (Employee_ID) في الجدول؛ فهو للدلالة على أن هذا الحقل يمثل أيضاً مفتاحاً خارجياً يشير إلى جدول «الموظف»؛ إضافةً إلى كونه جزءاً من المفتاح الرئيسي للجدول. وقد تمَّ إيضاح عملية الارتباط هذه (بين المفتاح الخارجي والمفتاح الرئيسي)، في الشكل رقم (5-2)، من خلال السهم الواصل بين الحقلين. ويحتوي كلُّ صف في جدول «مهارة الموظف» على رقم الموظف والمهارة التي تتوفر لديه.



شكل رقم (5–2): تحويل الخاصية المتعددة القيم إلى علاقة (أو جدول).

كما يُمكِن أن يقترح التمثيل السابق على المستفيدين من قاعدة البيانات؛ إضافةَ المزيد من الحقول إلى جدول «مهارة الموظف»، مثل: تاريخ اكتساب المهارة، أو مكان الحصول عليها... إلخ. وتجدر الإشارة هنا إلى أن الخاصية المتعددة القِيم قد تكون مركبةً أيضاً. فعلى سبيل المثال: من

الممكن تمثيلُ خاصية المهارة في النموذج المفاهيمي من الأساس على أنها تتكون من اسم المهارة، وتاريخ الحصول عليها، ومكان الحصول عليها. في مثل هذه الحالة؛ يتمُّ وَضْع المهارة (Skill) ضِمن شكلٍ بيضاوي مزدوج الخطوط (كما هو أعلاه)، ويتفرَّع منه بقية الخصائص البسيطة الثلاث موضوعة، كلٌّ على حِدة، ضِمن أشكال بيضاوية مفردة الخطوط كما لو كنَّا نحاول نمذجة خاصية مركبة. وعند ارتباط خاصية مركبة متعددة القيم؛ يتمُّ تمثيل خصائصها البسيطة فقط ضِمن الجدول الثاني؛ إضافةً إلى حقل المفتاح الرئيسي للجدول الأول، كما هو مُوضَّح في الشكل رقم (5-3)، على افتراض أن اسم المهارة يُعَدُّ مميزاً للمهارات التي يتمتع بها الموظفون.



شكل رقم (5–3): تحويل الخاصية المركبة المتعدِّدة القيم إلى علاقة (أو جدول).

وفي حالة ارتباط أكثر من خاصية واحدة متعددة القيم بـكينونةٍ ما؛ فإنه يتمُّ إنشاء جدول لكلِّ واحدة من الخصائص المتعددة القيم. ويكون المفتاح الرئيسي للجدول الذي يمثل الكينونة الرئيسية جزءاً من المفاتيح الرئيسية للجداول التي تمثل الخصائص المتعددة القيم. كما يكون جزءُ المفتاح الرئيسي للكينونة الرئيسية مفتاحاً خارجياً في كلِّ جدول من جداول الخصائص المتعددة القيم يشيرُ إلى الجدول الرئيسي. أمَّا الخاصية المتعددة القيم نفسها، في كلِّ جدول، فتمثِّل الحقل الثاني في الجدول، وتكون جزءاً من مفتاحه الرئيسي.

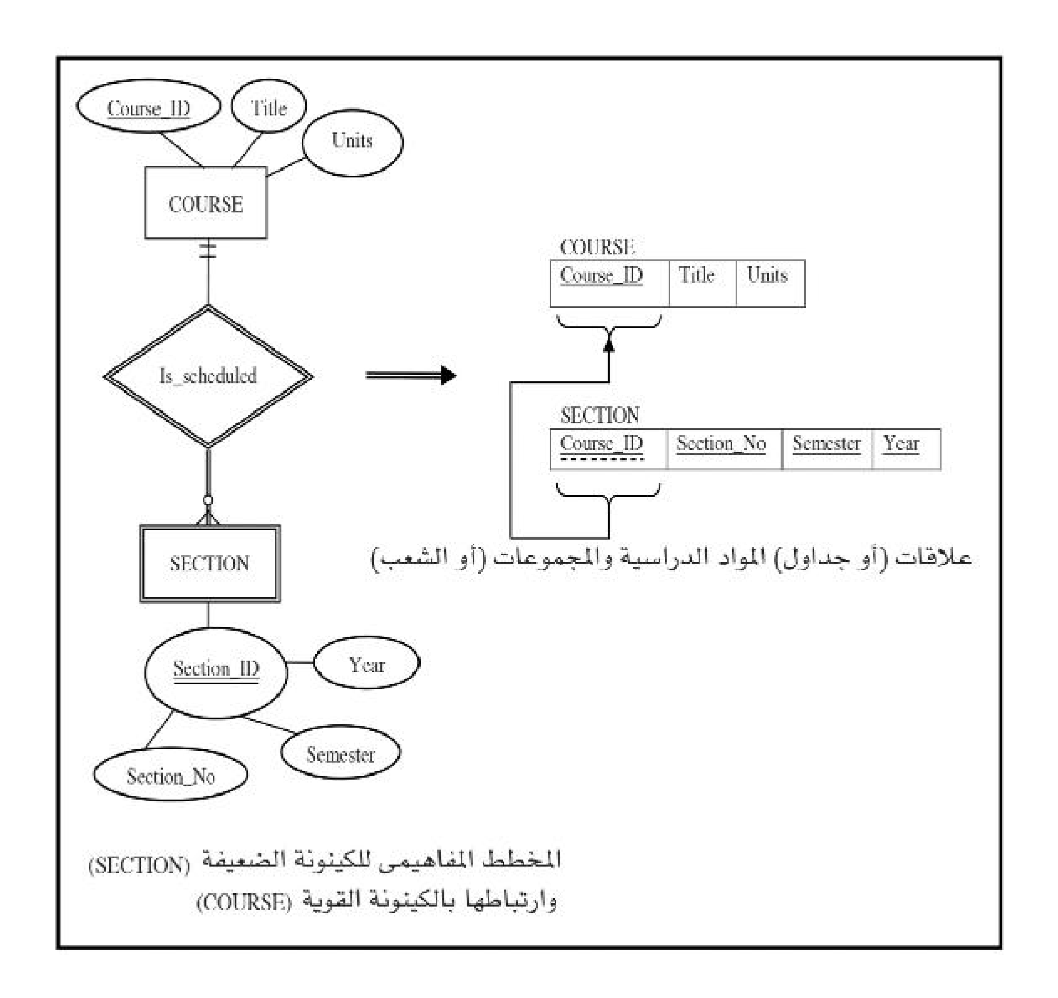
5–1–2 قاعدة التحويل الثانية: التعامل مع الكينونات الضعيفة:

الكينونات الضعيفة؛ هي تلك الكينونات التي لا يمكن أن تُوجَد في النموذج المفاهيمي باستقلالية؛ لكونها تعتمد على علاقات معرفة تربطها بـكينونات قوية. كما أن أيَّ كينونة ضعيفة لا يتوفر لديها مُعرِّف كامل يميِّز بين حالاتها المختلفة؛ غير أنه لا بد أن يتوفر لديها مُعرِّف جزئي يتكوَّن من خاصية (أو أكثر) تستطيع أن تميِّز بين مجموعة الحالات التي ترتبط بـكلِّ حالة من حالات الكينونة القوية. والشكل رقم (5-4) يُوضِّح ارتباط الكينونة الضعيفة وهي كينونة «المجموعة الدراسية» (SECTION)؛ من خلال العلاقة المعرَّفة، وهي «جدولة المجموعة» (Is_sCheduled) بالكينونة القوية، وهي «المادة الدراسية» (COURSE). ومعنى هذا أنه لا يمكن أن توجَد مجموعة دراسية دون أن ترتبط بـمادة دراسية معينة ضِمن كينونة «المادة الدراسية». كما أن الخصائص البسيطة المكونة للخاصية المركبة «رمز المجموعة» (SeCtion_ID) يمكننا من التمييز بين المجموعات الدراسية التابعة لأيِّ مادة دراسية؛ ولكنها لا تستطيع أن تميِّز بين المجموعات التابعة لمواد دراسية مختلفة؛ لكونها قد تأخذ القيم نفسها. لذلك؛ فإن الخاصية المركبة تُعَدُّ مُميِّزاً جزئياً للكينونة الضعيفة، وبالتالي؛ فإننا نستخدم، عند تحويل الكينونة الضعيفة لجدول علاقي، الخاصية المعرفة للكينونة القوية بالإضافة إلى المعرِّف الجزئي للكينونة الضعيفة في تعريف المفتاح الرئيسي للجدول.

وتتمُّ عملية تحويل أيِّ كينونة ضعيفة إلى النموذج العلاقي؛ من خلال إنشاء جدول يحتوي على حقول لكافة الخصائص البسيطة المرتبطة بالكينونة، كما لو أننا نقوم بـتحويل كينونة قوية إلى

جدول علاقي. إضافةً إلى ذلك؛ يتمُّ إدراج حقل في الجدول لخاصية معرف الكينونة القوية. ويكون المفتاح الرئيسي لجدول الكينونة الضعيفة عبارة عن حقل (أو حقول) المعرِّف الجزئي للكينونة الضعيفة؛ إضافةً إلى حقل (أو حقول) المفتاح الرئيسي للكينونة القوية. فعلى سبيل المثال: عند تحويل كينونة «المجموعة الدراسية» إلى النموذج العلاقي؛ يتمُّ إنشاء جدول بمُسمَّى الكينونة نفسه ويحتوي على حقول لتمثيل جميع الخصائص البسيطة المرتبطة بالكينونة، وهي: «رقم المجموعة»، و«الفصل» الدراسي المنفذة فيه، و«السنة» الدراسية المنفذة فيها. ولكون هذه الحقول الثلاثة مجتمعةً تُعَدُّ معرفاً جزئياً لا يمكِّننا من التمييز بين المجموعات التابعة لمواد دراسية مختلفة؛ فقد تمَّ إدراجُ حقل لتمثيل معرِّف الكينونة القوية وهو «رمز المادة الدراسية»، وفي ذات الوقت تمَّ تعريف هذا الحقل على أساس أنه جزءٌ من المفتاح الرئيسي للجدول.

وبهذه الطريقة يمكننا الآن التمييز بين جميع المجموعات الدراسية، بشكلٍ منفردٍ، بغض النظر عن المادة الدراسية التي تتبع لها. ولأن وجود كينونة ضعيفة يعني دائماً وجود علاقة بينها وبين الكينونة القوية التي ترتكز عليها، وأن هذه العلاقة لا يمكن أن تكون «متعدد - متعدد» (لأن كلَّ حالة من حالات الكينونة الضعيفة لا يمكن أن ترتبط بأكثر من حالةٍ من حالات الكينونة القوية)؛ فإنه يتمُّ تمثيل هذه العلاقة من خلال تعريف حقل «رمز المادة الدراسية» في جدول المجموعات الدراسية على أنه مفتاح خارجي يشير إلى المفتاح الرئيسي في جدول المواد الدراسية. وقد تمَّ إيضاح ذلك في الشكل رقم (5-4) من خلال وَضْع خط متقطع تحت حقل «رمز المادة الدراسية»؛ إضافةً إلى استخدام سهم يوضِّح عملية الارتباط هذه بين الجدولين. وسيتمُّ شرح طرق تحويل العلاقات بشكلٍ أكثر تفصيلاً في الأجزاء التالية. تجدرُ الملاحظة أنه لم يتم إدراج خاصية «الموقع» (LoCation) وربطها بالمجموعة الدراسية ضِمن المخطط المفاهيمي في الشكل رقم (5-4) أو إدراج ما يقابلها من حقل في جدول «المجموعة الدراسية» في المخطط المنطقي؛ وذلك فقط حتى لا يكتظ الشكل بالرسومات.



شكل رقم (5–4): تحويل الكينونة الضعيفة إلى علاقة (أو جدول).

5–1–3 قاعدة التحويل الثالثة: التعامل مع العلاقات الثنائية:

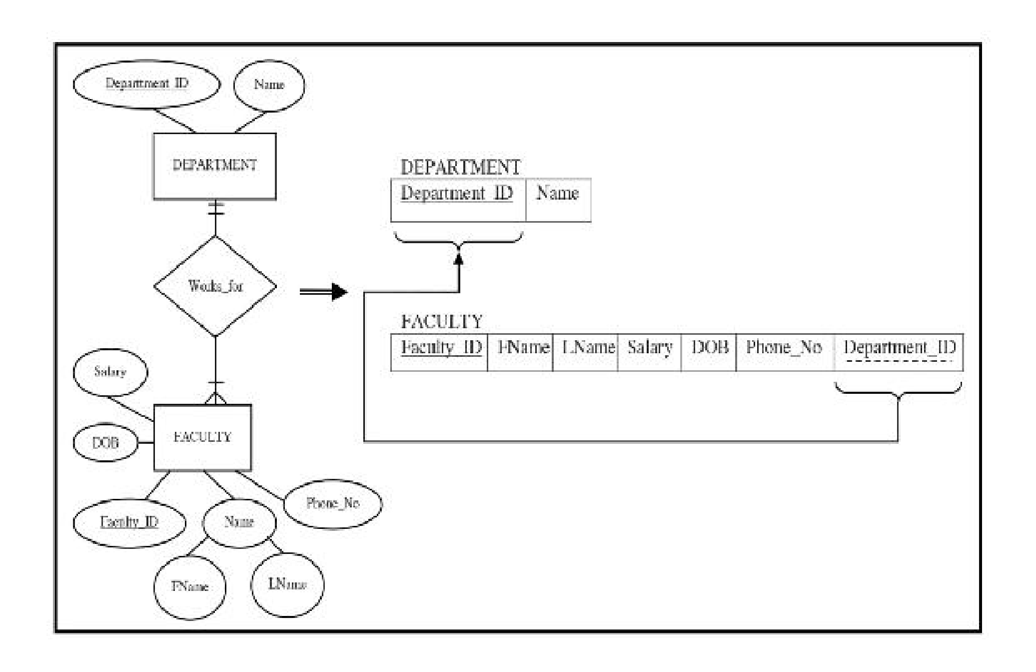
تعتمد عملية تحويل العلاقات على درجاتها؛ من حيث كونها أحادية، أو ثنائية، أو ثلاثية (فأكثر). كما أن عملية التحويل تعتمد أيضاً على تعدُّدية العلاقة؛ من حيث كونها «واحد - واحد»، أو «واحد - متعدد»، أو «متعدد - متعدد». وبخلاف تعدُّدية العلاقة في النموذج المفاهيمي «كينونة - علاقة»؛ فإننا ننظر فقط إلى التعدُّدية العليا للعلاقة في مرحلة التصميم المنطقي، دون النظر إلى التعدُّدية الدنيا لها في هذه المرحلة؛ غير أنه من الضروري الإشارة إلى أن التعددية الدنيا ذات أهميةٍ كبيرة في أثناء عملية بناء هياكل قاعدة البيانات كما سنوضِّح عند شرحنا للغة الاستفسار

البنائية في الفصلين السابع والثامن. وتوضِّح الأجزاء التالية الطرق المتبعة لتحويل العلاقات حسب درجاتها وتعدُّدياتها.

5–1–3–1 التعامل مع العلاقات الثنائية ذات التعددية «واحد – متعدد»:

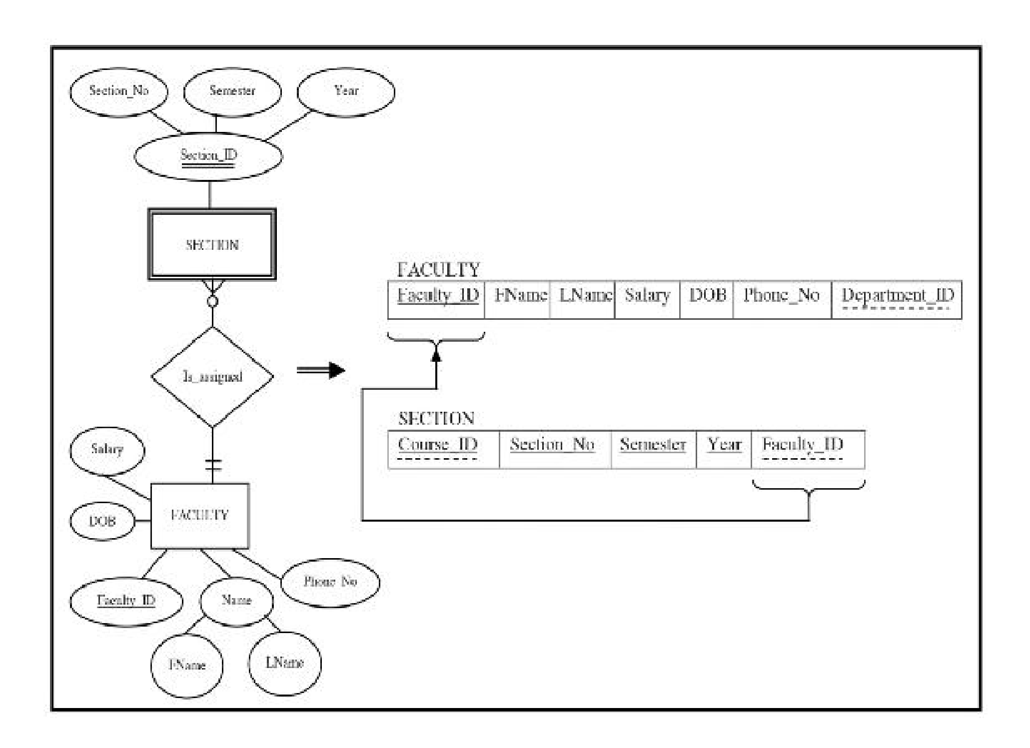
عند وجود علاقة ذات درجة ثنائية (تربط بين كينونتين) وتعدُّدية «واحد - متعدد»؛ يتمُّ أولاً إنشاءُ جدول لكلِّ كينونة من الكينونات المرتبطة بالعلاقة الثنائية، وفقاً للقاعدة الأولى أعلاه. بعد ذلك يتمُّ إدراج المفتاح الرئيسي (سواء كان حقلاً واحداً أو أكثر) للجدول الذي يمثل الكينونة في الجانب ذي التعدُّدية «واحد» ضمن حقول الجدول الذي يمثل الكينونة في الجانب ذي التعدُّدية «متعدد». ويتمُّ تعريف هذا الحقل (أو الحقول) باعتباره مفتاحاً خارجياً يشيرُ إلى الجدول الذي يمثل الكينونة في الجانب ذي التعدُّدية «واحد».

ولإيضاح عملية التحويل هذه؛ لنأخذ على سبيل المثال العلاقة الثنائية «يعمل في» (Works_for) التي تربط بين كينونة «القسم الدراسي» (DEPARTMENT) وكينونة «عضو هيئة التدريس» (FACULTY) في نموذج «كينونة - علاقة» للجامعة الأهلية الموضَّحة في الشكل رقم (5-5). فهذه العلاقة؛ إضافةً إلى كونها علاقة ثنائية؛ فهي علاقة ذات تعددية «واحد - متعدد»؛ وذلك لكون كلِّ عضو هيئة تدريس يعمل في قسم دراسي واحد (على الأكثر) وأن كلَّ قسم دراسي يعمل فيه أكثر من عضو هيئة تدريس. ولأن التعدُّدية «واحد» في هذا النموذج تأتي في جانب كينونة «القسم الدراسي»؛ فإنه يتمُّ إدراج حقل جديد في جدول أعضاء هيئة التدريس، ذي الجانب المتعدد؛ لتمثيل المفتاح الرئيسي لجدول الأقسام العلمية. ويتمُّ تعريف هذا الحقل على أنه مفتاح خارجي. وقد تمَّ إيضاح ذلك في عملية التحويل من خلال وضع خط متقطع تحت مُسمَّى هذا الحقل، كما تمَّ وَضْع سهم يشير إلى هذا الارتباط. وبهذه الطريقة يمكننا دائماً معرفة القسم الذي يعمل فيه كلُّ عضو من أعضاء هيئة التدريس. ونظراً لكون التعددية إجبارية؛ فإن قيمة حقل رمز القسم الدراسي في جدول أعضاء هيئة التدريس لا يمكن أن تكون غائبةً (NULL). وعلى الرغم من عدم إمكانية فرض هذا القيد على حقل المفتاح الخارجي في هذه المرحلة من التصميم؛ فإنه يمكِن فرضه في أثناء مرحلة بناء قاعدة البيانات باستخدام قيد القيم الغائبة (NOT NULL)، الذي يُعَدُّ أحد القيود التي من الممكن أن تُفرَض على الحقول، كما سنوضِّحه في الفصل السابع.



شكل رقم (5–5): تحويل العلاقة الثنائية ذات التعدُّدية «واحد – متعدد» إلى النموذج العلاقي.

وقد يُطرَح السؤال التالي: هل من الممكن وَضْع المفتاح الرئيسي لجدول أعضاء هيئة التدريس باعتباره مفتاحاً خارجياً في جدول الأقسام الدراسية لتمثيل العلاقة الثنائية السابقة في النموذج العلاقي عوضاً عن تمثيلها بالطريقة السابقة؟ إنَّ الإجابة عن التساؤل؛ هي عدم إمكانية ذلك، والسبب يرجع إلى أنه لو فعلنا ذلك؛ فإننا سنقوم بتكرار قيمة المفتاح الرئيسي واسم القسم لجميع أعضاء هيئة التدريس الذين يعملون في القسم الدراسي نفسه؛ وبذلك لن يصبح حقل ««رمز القسم»»، في جدول الأقسام الدراسية مفتاحاً رئيسياً؛ لكونه يتكرَّر في سجلات الجدول. كما أن عملية تكرار البيانات هذه قد تؤدي إلى إشكالات (Anomalies) عند التعديل على البيانات يَصعُب في ظل وجودها التحكُّم في تناسق البيانات (Data ConsistenCy)، كما سيتضح عند شرح الجداول جيدة البناء وعمليات تطبيع الجداول (Normalization) (في الجزء الأول من الفصل السادس).



شكل رقم (5–6): تحويل العلاقة الثنائية ذات التعددية «واحد – متعدد» ترتبط بها كينونة ضعيفة إلى النموذج العلاقي.

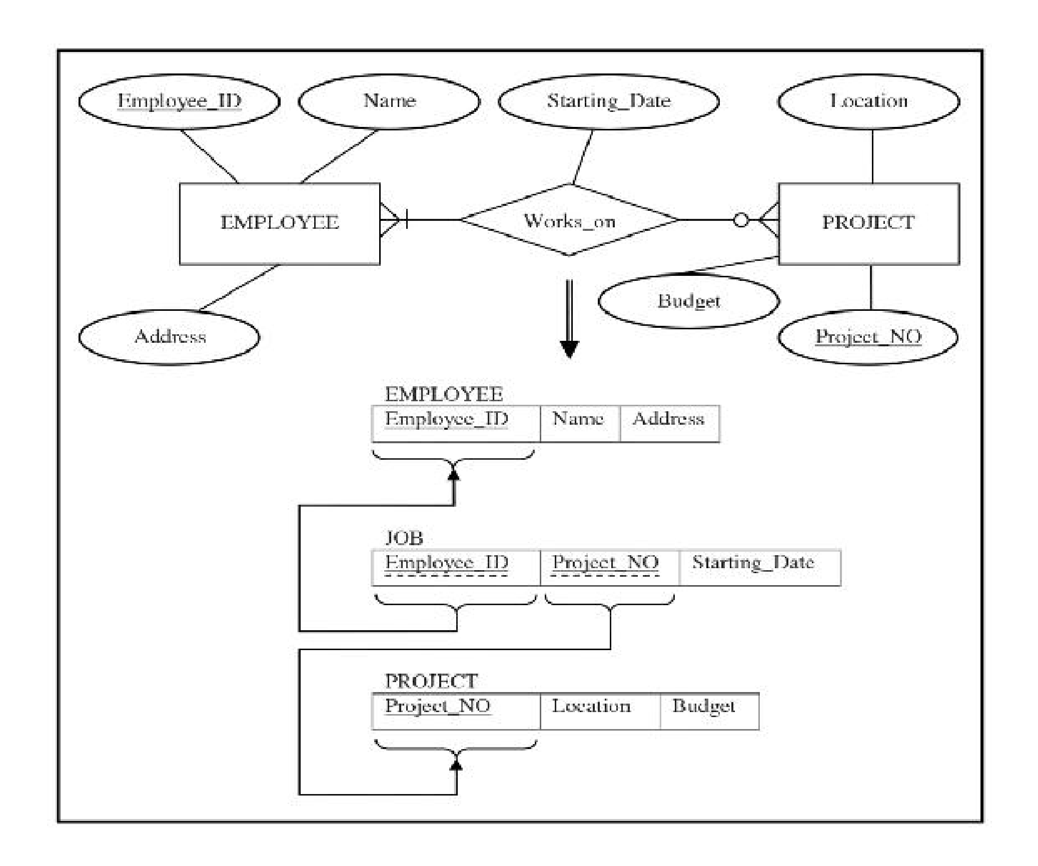
أمَّا الشكل رقم (5-6)؛ فيمثل علاقة ثنائية «واحد - متعدد» تربط بين كينونة قوية، وهي كينونة «عضو هيئة التدريس» (FACULTY)، وكينونة ضعيفة هي كينونة «المجموعة الدراسية» (SECTION). وتعني هذه العلاقة، الممثلة في النموذج، أن كلَّ عضو هيئة تدريس قد يدرس مجموعةً دراسيةً أو أكثر في حين تدرس كلُّ مجموعة دراسية من قِبل عضو هيئة تدريس واحد فقط. ونظراً لأن العلاقة بين الكينونتين ليست علاقة معرفة (Identifying Relationship)؛ بمعنى أن كينونة «عضو هيئة التدريس» ليست الكينونة التي تميِّز بين المجموعات الدراسية، وإنما كينونة «المادة الدراسية» هي الكينونة القوية التي تميِّز بين المجموعات المختلفة؛ فإنه يتم التعامل مع الكينونة الضعيفة كأنها كينونةً قويةً عند عملية التحويل للنموذج العلاقي؛ وذلك حسب القاعدة السابقة. ففي هذه الحالة؛ يتمُّ إدراج حقل جديد ضمن جدول المجموعات الدراسية (وهو الجانب المتعدد) لتمثيل المفتاح الرئيسي لجدول أعضاء هيئة التدريس، كما يتمُّ تعريف هذا الحقل على أنه مفتاح خارجي؛ من خلال وَضْع خط متقطع تحت مسماه. كما يُلاحَظ وجود حقول أخرى في الشكل قد تمَّ وَضْع خطوط متقطعة تحت مسمياتها، سواء في جدول أعضاء هيئة التدريس أم في جدول المجموعات الدراسية. وهذه الحقول تظهرُ بهذا الشكل؛ للدلالة

على أنها مفاتيح خارجية تمثِّل علاقات أخرى مع هذين الجدولين تمَّ تحويلها في مراحل سابقة. تجدر الملاحظة في الشكل رقم (5-6) أيضاً أنه لم يتم إدراج خاصية «الموقع» (LoCation)، وربطها بالمجموعة الدراسية ضمن المخطط المفاهيمي أو إدراج ما يقابلها من حقل في جدول «المجموعة الدراسية» في المخطط المنطقي؛ وذلك فقط حتى لا يكتظ الشكل بالرسومات.

5–1–3–2 التعامل مع العلاقات الثنائية ذات التعدُّدية «متعدد – متعدد»:

عند وجود علاقة ثنائية ذات تعدُّدية «متعدد - متعدد» تربط بين كينونتين؛ فإننا نقوم بإنشاء ثلاثة جداول، اثنان منها لتمثيل كلِّ كينونة على حِدَة (حسب القاعدة (1) أعلاه)، وجدول ثالث لتمثيل العلاقة نفسها؛ بحيث يتضمَّن جدول العلاقة حقلين يمثلان المفاتيح الرئيسية لجدولي الكينونتين؛ إضافةً إلى الحقول اللازمة التي تمثل الخصائص المرتبطة بـالعلاقة نفسها (عند وجود مثل هذه الخصائص). ويصبح حقلا المفاتيح الرئيسية لجدولي الكينونتين التي تربط بينهما العلاقة مجتمعين المفتاح الرئيسي للجدول الذي يمثل العلاقة، كما يتمُّ تعريف كلٍّ منهما باعتباره مفتاحاً خارجياً يشيرُ إلى الجدول الذي تمَّ جلب الحقل منه.

ويمثل الشكل رقم (5-7) علاقةً ثنائيةً ذات تعددية «متعدد - متعدد»، وهي علاقة «يعمل على» (Works_on) تربط بين كينونتين هما: كينونة «الموظف» (EMPLOYEE) وكينونة «المشروع» (PROJECT). ويمكِن أن تُقرأ هذه العلاقة كما يلي «يعمل كلُّ موظف على صفر أو أكثر من المشاريع، وكل مشروع يعمل عليه موظفٌ واحد أو أكثر. وعندما يعمل موظف ما على مشروع معين؛ فإن هناك تاريخاً يبيِّن بداية عمل الموظف على المشروع». ونظراً لأن تاريخ العمل على المشروع ليس من خصائص أيٍّ من كينونة «الموظف» أو كينونة «المشروع»، وإنما هي من خصائص العلاقة التي تربط بينهما؛ فقد تم ربطها بالعلاقة وليس بـأيٍّ من الكينونتين. وحسب قاعدة التحويل أعلاه؛ يتمُّ إنشاء ثلاثة جداول: جدولان يمثلان الكينونتين التي تربط بينهما العلاقة، وهما جدول «الموظف» وجدول «المشروع»، وجدول ثالث يمثل العلاقة نفسها، وقد سُمِّي جدول «العمل» (JOB). وقد تمَّ تغييرُ مُسمَّى الجدول هنا عن مُسمَّى العلاقة حتى يتوافق مُسمَّى جدول العلاقة (أو جدول الربط) مع قواعد تسمية الكينونات والجداول التي يُفضَّل أن تكون أسماء عوضاً عن تسميتها بـأفعال.



شكل رقم (5–7): تحويل العلاقة الثنائية ذات التعدُّدية «متعدد – متعدد» إلى النموذج العلاقي.

ويوضِّح الشكل رقم (5-7) أيضاً الحقول المكوِّنة للجدولين اللذين يمثلان الكينونتين اللتين تربط بينهما العلاقة؛ إذ تمَّ تعريفهما حسب قاعدة التحويل رقم (1) أعلاه. أما فيما يتعلق بـجدول العلاقة نفسها؛ فقد تمَّ تعريف حقلين فيه؛ أحدهما لتمثيل المفتاح الرئيسي لجدول الموظفين، والثاني لتمثيل المفتاح الرئيسي لجدول المشاريع. كما تمَّ تعريف حقلٍ ثالث في جدول العلاقة؛ لتمثيل خاصية «بداية العمل» (Starting_Date) المرتبطة بـالعلاقة نفسها. ويُلاحَظ في جدول العلاقة وَضْع خط متصل تحت الحقلين اللذين يمثلان المفاتيح الرئيسية لكلٍّ من جدول الموظفين وجدول المشاريع؛ وذلك للدلالة على أن كليهما مجتمعين يمثلان المفتاح الرئيسي لجدول العلاقة. وإضافةً إلى ذلك؛ فقد تمَّ وَضْع خط متقطع تحت كلٍّ منهما؛ للدلالة على أن كلَّ واحدٍ من الحقلين يمثل مفتاحاً خارجياً يشيرُ إلى الجدول الذي جُلِب منه. ولإيضاح ذلك؛ فقد تمَّ استخدام أسهم تصل بين كلِّ مفتاح خارجي بـحقل الجدول الذي يشير إليه المفتاح.

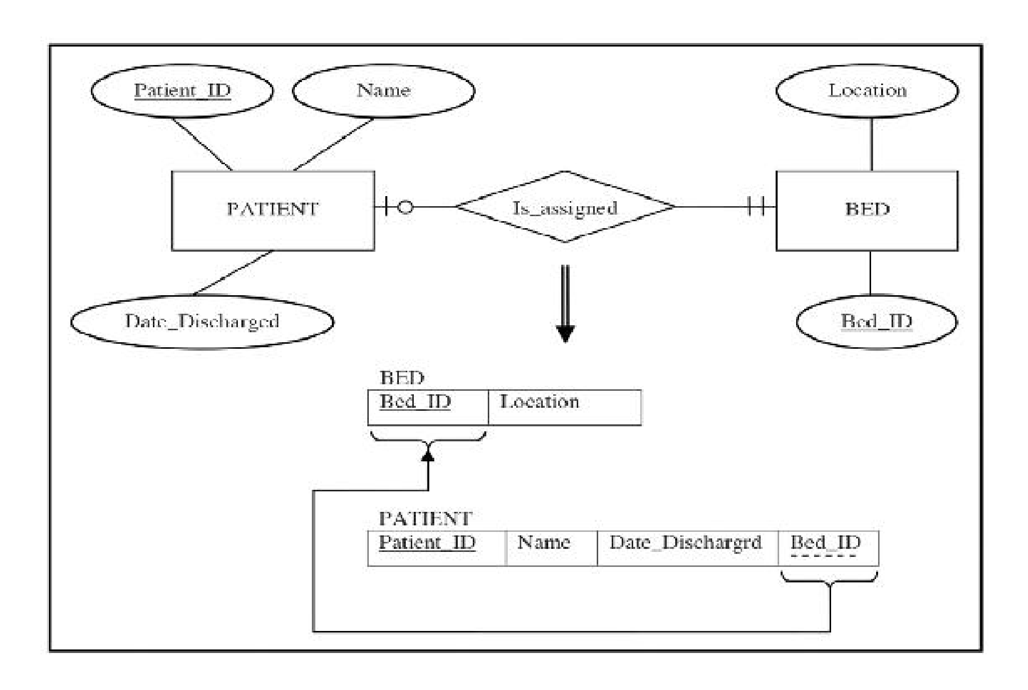
غير أنه قد يُطرَح السؤال التالي: لماذا لا يتم تمثيلُ العلاقة الثنائية «متعدد - متعدد» من خلال إضافة حقل لأحد جدولي الكينونتين، ويصبح هذا الحقل مفتاحاً خارجياً في الجدول الذي تمَّت

إضافته إليه (لتمثيل العلاقة) كما هو الحال في العلاقات الثنائية ذات التعددية «واحد - متعدد»؛ عوضاً عن تعريف جدول ثالث خاص بالعلاقة نفسها؟ والإجابة عن ذلك أنَّ هذه الطريقة لا تمكِّن من تمثيل العلاقات الثنائية ذات التعددية «متعدد - متعدد». والسبب وراء ذلك؛ هو أننا لا نعرف الحدَّ الأعلى من عدد المشاريع التي من الممكن أن يعمل عليها الموظف الواحد، أو الحدَّ الأعلى من عدد الموظفين الذين من الممكن أن يعملوا على المشروع الواحد. ونتيجةً لذلك لا نستطيع تعريف عددٍ مُحدَّدٍ من الحقول، سواء في جدول الموظفين أم في جدول المشاريع لتمثيل العلاقة. وحتى لو عُرِف الحدُّ الأعلى من عدد المشاريع أو الحدُّ الأعلى من عدد الموظفين؛ فإن مثل هذا التمثيل سيضيع الكثير من مساحة التخزين؛ لأن عدد الموظفين الذين يعملون على كلِّ مشروع (أو عدد المشاريع التي يعمل عليها كل موظف) قد تتفاوت بشكلٍ كبير. إضافةً إلى ذلك؛ فإنه يجب تمثيل تاريخ البدء في العمل على كلِّ مشروع من قِبل كلِّ موظف، وفق الحَدِّ الأعلى المستخدَم؛ مما سيزيد من حجم المساحة التخزينية المُهدَرة.

5-1-3-3 التعامل مع العلاقات الثنائية ذات التعددية «واحد - واحد»:

يمكِن النظرُ إلى العلاقات الثنائية ذات التعددية «واحد- واحد» على أنها حالةٌ خاصةٌ من العلاقات الثنائية «واحد - متعدد»؛ إذ يتمُّ تحويلها بخطوتين: في الخطوة الأولى يتمُّ إنشاء جدولٍ لكلٍّ من الكينونتين اللتين تربط بينهما العلاقة الثنائية. أمَّا في الخطوة الثانية؛ فيتمُّ تمثيل المفتاح الرئيسي لجدول إحدى الكينونتين باعتباره حقلاً في الجدول الذي يمثل الكينونة الأخرى، ويتمُّ تعريف الحقل المضاف على أنه مفتاح خارجي للجدول الذي يمثل الكينونة الأولى. ويُلاحَظ مدى تشابه هاتين الخطوتين مع قاعدة تحويل العلاقات الثنائية ذات التعددية «واحد - متعدد». إلا أن الاختلاف هنا يظهر عند تحديد الجدول الذي سيُضاف إليه المفتاح الخارجي، والذي سيمثل العلاقة. ففي حالة التعدُّدية «واحد - متعدد»؛ يتم إدراج المفتاح الرئيسي للجدول الممثل لكينونة الجانب الأحادي، من العلاقة، في الجدول الممثل لكينونة الجانب المتعدِّد على أن يكون مفتاحاً خارجياً للجدول الممثل للجانب ذي التعددية الأحادية. أمَّا هنا؛ فيتمُّ إدراج المفتاح الرئيسي للجدول الممثل لكينونة الجانب الإجباري، من العلاقة، في الجدول الممثل لكينونة الجانب الاختياري على أن يكون مفتاحاً خارجياً للجدول الممثل للجانب الاختياري. وكما أسلفنا سابقاً؛ فإن ارتباط أية كينونة بعلاقة إما أن يكون إجبارياً أو اختيارياً. ويُمثَّل هذا ضِمن النموذج المفاهيمي «كينونة - علاقة» بالقيمة

الصغرى (أو الدنيا). فإذا كانت القيمة الصغرى صفراً؛ فإن العلاقة تُعَدُّ اختياريةً. أما إذا كانت القيمة الصغرى واحداً؛ فإنها تُعَدُّ إجباريةً.



شكل رقم (5–8): تحويل العلاقة الثنائية ذات التعددية «واحد – واحد» إلى النموذج العلاقي.

وفي العلاقات الثنائية ذات التعدُّدية «واحد - واحد»؛ تكون التعدُّدية في غالبية الأحيان إجباريةً من جانب واختياريةً من الجانب الآخر. فعلى سبيل المثال: يُوضِّح الشكل رقم (5-8) كينونة «مريض» (PATIENT) وكينونة «سرير» (BED) اللتين ترتبط إحداهما بالأخرى؛ من خلال العلاقة الثنائية «واحد - واحد» وهي علاقة «يُسنَد إلى» (Is_assigned). ويعني التمثيل الموضَّح في الشكل أن «كلَّ مريض في المستشفى يجب أن يُسنَد إلى سرير واحد فقط، في حين قد يُسنَد السرير لمريضٍ ما، أو قد لا يُسنَد لأيِّ مريض». وبناءً على هذا التمثيل؛ فإن الجانب الإجباري هو من جهة السرير؛ إذ إن كلَّ مريضٍ لا بد أن يُسنَد إلى سرير. وباتباع القاعدة أعلاه؛ يتمُّ إنشاءُ جدولين أحدهما لتمثيل كينونة «المريض» والآخر لتمثيل كينونة «السرير». كما تتمُّ إضافة حقلٍ جديدٍ في جدول الجانب الاختياري من العلاقة، وهو جدول «المريض»؛ لتمثيل المفتاح الرئيسي لجدول الجانب الإجباري، وهو «السرير». كما يتمُّ تعريف الحقل الذي تمَّت إضافته على أنه مفتاح خارجي يشيرُ إلى جدول الجانب الإجباري، وهو «السرير». وفي حالِ ارتبطت العلاقة نفسها بخصائص؛ فإنها تُضاف أيضاً باعتبارها حقولاً ضِمن الجانب الاختياري

من العلاقة. فلو ارتبطت علاقة إسناد السرير في مثالنا بخاصية مثل «تاريخ بداية الإسناد» (Starting_Date)؛ فإنه يتمُّ تمثيل هذه الخاصية باعتبارها حقلاً مصاحباً للمفتاح الخارجي ضِمن جدول «المريض».

والسؤال الذي قد يُطرَح هو: هل بالإمكان تمثيلُ العلاقة الثنائية ذات التعدُّدية «واحد - واحد» بالشكل المعاكس؟ بمعنى: هل من الممكن أن يُعرَّف المفتاح الرئيسي لجدول الجانب «الاختياري» (وهو المريض في مثالنا) على أنه مفتاح خارجي ضِمن جدول الجانب الإجباري (وهو «السرير» في مثالنا؛ إضافةً لأية خصائص قد تكون مرتبطةً بالعلاقة نفسها)؟ إن الإجابة عن هذا التساؤل هي: نعم يمكننا ذلك؛ ولكنها ليست الطريقة المثلى. والسبب وراء ذلك؛ أنه ليس من الضروري أن يكون «السرير» مسنداً لأيِّ مريض. وفي هذه الحالة؛ ستكون قيمة حقل المفتاح الخارجي (وبقية الحقول المرتبطة بالعلاقة إن وُجِدت) غائبةً (NULL)؛ للدلالة على أن السرير غير مُسنَد إلى أيِّ مريض. ويعني هذا إهدار المساحة التخزينية في حالات عدم الإسناد هذه.

على الرَّغم من أن الحالة العامة للعلاقات الثنائية ذات التعدُّدية «واحد- واحد» هي وجود جانب إجباري وجانب اختياري؛ غير أنه من الممكن أن يكون كلا الجانبين إجباريين أو كلاهما اختياريين. ففي الحالة الأولى (وهي كون العلاقة إجبارية من الجهتين) تتمُّ عملية تحويل العلاقة حسب الخطوتين السابقتين؛ وبحيث يتمُّ تعريف حقل المفتاح الرئيسي لأيٍّ من الجدولين على أنه مفتاح خارجي ضِمن الجدول الآخر (إضافةً إلى الحقول المرتبطة بالعلاقة) دون أيَّة أفضلية بين الجدولين. أمَّا في الحالة الثانية (وهي كون العلاقة اختيارية من الجهتين)، وعلى الرَّغم من إمكانية استخدام طريقة التحويل السابقة نفسها؛ فإنَّ عملية التحويل يمكِن أن تتمَّ كما لو أننا نقوم بتحويل علاقة ثنائية ذات تعددية «متعدد - متعدد»، بمعنى إنشاء جدول ثالث لتمثيل العلاقة عوضاً عن تضمينها ضِمن أحد الجدولين. وتأتي أهمية التمثيل الأخير للعلاقات الثنائية ذات التعددية «واحد - واحد» عند كون حالات عدم ارتباط الكينونتين التي تربط بينهما العلاقة هي الحالة العامة؛ إذ إنه سيتمُّ تجنُّب الكثير من القِيم الغائبة مقارنةً بالطريقة السابقة.

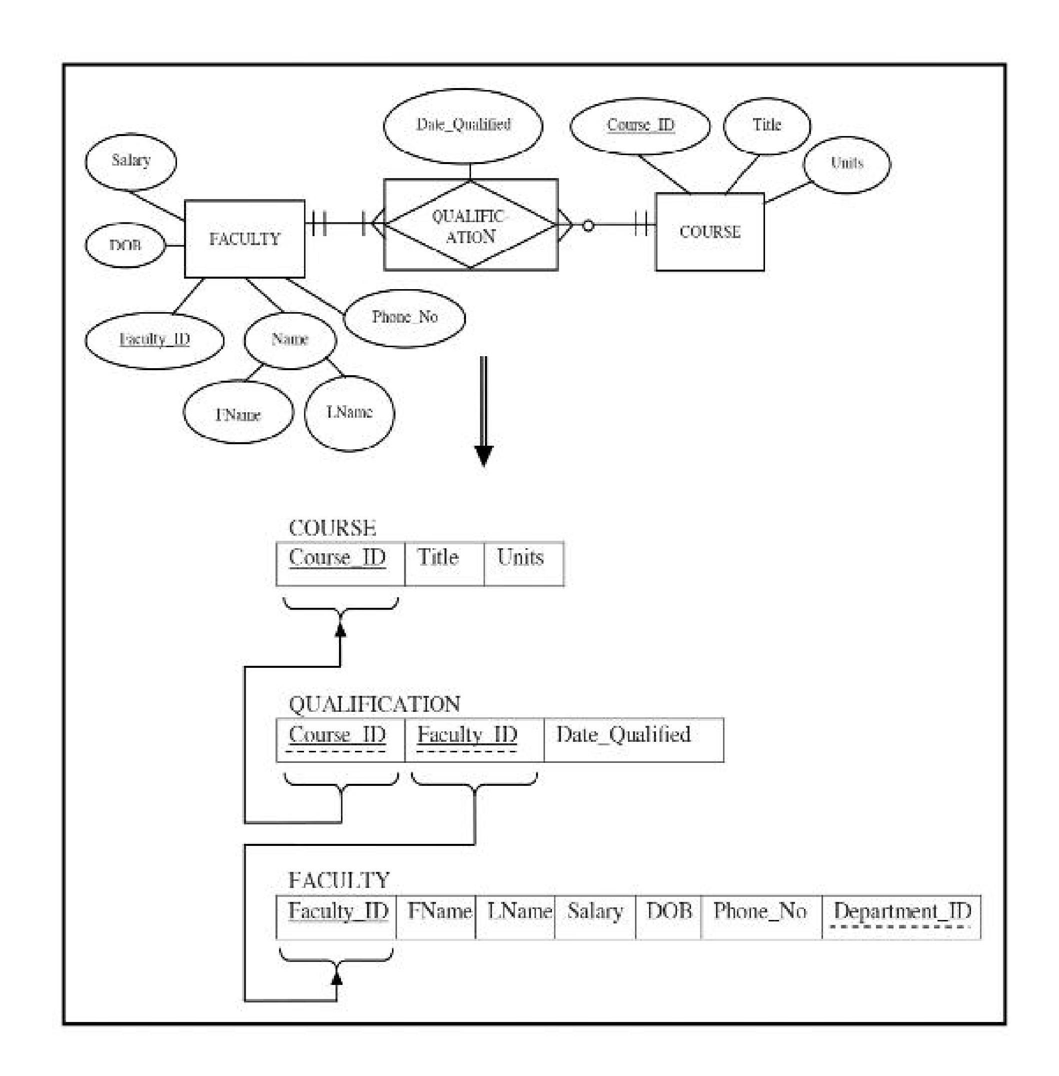
5-1-4 قاعدة التحويل الرابعة: التعامل مع الكينونات المشاركة:

إنَّ الكينونة المشاركة هي في أصلها علاقةٌ ذات تعدُّدية «متعدد – متعدد»؛ ولكن الشخص الذي يقوم بتصميم نموذج «كينونة - علاقة» قد يرى أنه من الأنسب تمثيل هذه العلاقة باعتبارها كينونة مشاركة؛ وذلك عندما يكون تمثيلها بهذه الطريقة أقرب إلى فهم المستفيدين من قاعدة البيانات؛ عوضاً عن تمثيلها كعلاقة ذات تعددية «متعدد - متعدد». أمَّا عملية تحويل الكينونة المشاركة من النموذج المفاهيمي إلى النموذج العلاقي؛ فهي مماثلةٌ لعملية تحويل العلاقات ذات التعدُّدية «متعدد - متعدد». تتكوَّن عملية التحويل من خطوتين. في الخطوة الأولى؛ يتمُّ تعريف ثلاثة جداول: اثنان منها لتعريف الكينونتين اللتين تربط بينهما الكينونة المشاركة، والثالث لتعريف الكينونة المشاركة نفسها. أمَّا الخطوة الثانية؛ فتعتمد على وجود مُعرِّف للكينونة المشاركة من عدم وجود مُعرِّف لها، كما يلي:

5–1–4–1 التعامل مع الكينونات المشاركة عند عدم وجود مُعرِّف:

عند عدم ارتباط الكينونة المشاركة بخاصية معرِّفة تميِّز بين حالات الكينونة المشاركة؛ يتمُّ استخدامُ المفاتيح الرئيسية للجدولين اللذين يمثلان الكينونتين اللتين تربط بينهما الكينونة المشاركة مجتمعين باعتبارها مفتاحاً رئيسياً لجدول الكينونة المشاركة. وفي الوقت نفسه؛ يتمُّ تعريف كلِّ واحدٍ من هذين المفتاحين الرئيسيين على أنه مفتاحٌ خارجي يشيرُ إلى جدول الكينونة الذي جلب منه. وبهذه الطريقة تأتي عملية تحويل العلاقة المشاركة متطابقةً مع عملية تحويل العلاقات ذات التعددية «متعدد - متعدد».

ويوضِّح الشكل رقم (5-9) عملية تحويل إحدى الكينونات المشاركة الموجودة في نموذج «كينونة - علاقة» للجامعة الأهلية. ويحتوي الشكلُ على كينونة مشاركة هي علاقة «تأهيل» (QUALIFICATION) تربط بين كينونة «عضو هيئة التدريس» (FACULTY) وكينونة «المادة الدراسية» (COURSE). كما يرتبط بالعلاقة المشاركة خاصية واحدة؛ هي خاصية «تاريخ التأهيل» (Date_Qualified) تحدِّد التاريخ الذي تمَّ فيه تأهل عضو هيئة التدريس لتدريس مادة ما.



شكل رقم (5–9): تحويل الكينونة المشاركة عند عدم وجود معرف إلى النموذج العلاقي.

وباتباع خطوتي التحويل أعلاه؛ يتمُّ إنشاء ثلاثة جداول، هي: جدول أعضاء هيئة التدريس (FACULTY)؛ لتمثيل كينونة «عضو هيئة التدريس»، وجدول المواد الدراسية (COURSE)؛ لتمثيل كينونة «المادة الدراسية»، وجدول تأهيل (QUALIFICATION)؛ لتمثيل الكينونة المشاركة «تأهيل». ونظراً لعدم ارتباط الكينونة المشاركة بخاصية معرفة؛ فإنه يتمُّ تعريف المفتاح الرئيسي لجدول أعضاء هيئة التدريس، وهو «رمز عضو هيئة التدريس» (FaCulty_ID)، والمفتاح الرئيسي لجدول المواد الدراسية، وهو «رمز المادة الدراسية» (Course_ID)، مجتمعين، كمفتاح رئيسي للكينونة المشاركة. وفي الوقت نفسه؛ يتمُّ تعريف كلِّ جزءٍ من المفتاح الرئيسي في جدول الكينونة المشاركة على أنه مفتاح خارجي يشير إلى أحد جدولي الكينونتين اللتين تربط بينهما الكينونة المشاركة. فالحقل «رمز المادة الدراسية» (Course_ID) في جدول تأهيل (QUALIFICATION) يمثل جزءًا من المفتاح الرئيسي للجدول، وفي الوقت نفسه يمثل مفتاحاً خارجياً لجدول المواد الدراسية (COURSE). كذلك هو الحال بالنسبة لحقل «رمز عضو هيئة التدريس» (FaCulty_ID) الذي يمثل جزءًا من المفتاح الرئيسي لجدول تأهيل، وفي الوقت نفسه يمثل مفتاحاً خارجياً يشير إلى جدول أعضاء هيئة التدريس. وقد تمَّ إيضاح المفتاح الرئيسي لجدول تأهيل من خلال وَضْع خط متصلٍ تحت الحقلين الذين يتكوَّن منهما، وإيضاح المفاتيح الخارجية من خلال وَضْع خط متقطع تحت كلٍّ منهما. كما تمَّ إيضاح الجدول الذي يشير إليه كل مفتاح خارجي من خلال السهم الذي يصل بين المفتاح والجدول الذي يشير إليه المفتاح.

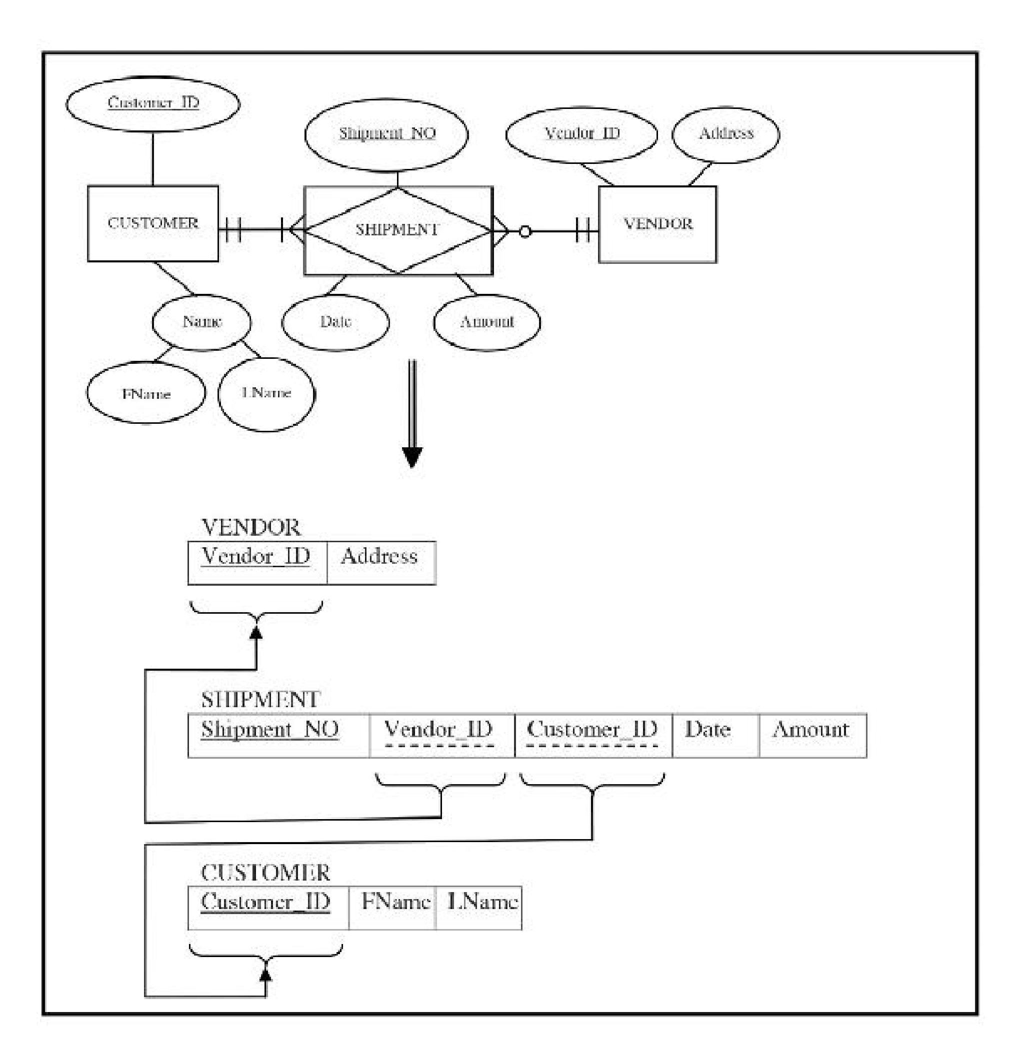
2-4-1-5 التعامل مع الكينونات المشاركة عند وجود معرف:

في بعض الأحيان تكون الكينونة المشاركة مرتبطةً بخاصية معرَّفة تميِّز بين حالات العلاقة المشاركة. وهناك سببان يُحفِّزان ربط الكينونة المشاركة بمعرِّف وهما (Hoffer et al, 2018):

1- قد يكون من الطبيعي وجود معرِّف للكينونة المشاركة معروف من قِبل المستخدِمين لقاعدة البيانات.

2- استخدام المفاتيح الرئيسية لجداول الكينونات التي تربط بينها الكينونة المشاركة على أنها مفتاح رئيسي في جدول الكينونة المشاركة؛ قد لا يُمكِّن من تمييز الحالات في جدول الكينونة المشاركة بـشكلٍ منفرد.

ولتحويل الكينونة المشاركة التي ترتبط بمعرِّف إلى النموذج العلاقي؛ يتمُّ تعريف جدول خاص بالكينونة المشاركة، كما سبق أعلاه، بالإضافة إلى تعريف جداول الكينونات التي تربط بـينها الكينونة المشاركة. غير أن وجه الاختلاف هنا يكمُن في أن المفتاح الرئيسي لجدول الكينونة المشاركة؛ سيكون معرف الكينونة عوضاً عن المفاتيح الرئيسية لجداول الكينونات التي تربط بـينها الكينونة المشاركة. أمَّا المفاتيح الرئيسية لجداول الكينونات التي تربط بينها الكينونة المشاركة؛ فيتم تعريفها ضِمن جدول الكينونة المشاركة باعتبارها مفاتيح خارجية فحسب.



شكل رقم (5–10): تحويل الكينونة المشاركة عند وجود معرف إلى النموذج العلاقي.

ويُوضِّح الشكل رقم (5-10) عملية تحويل إحدى الكينونات المشاركة التي ترتبط بخاصية معرفة. ويحتوي الشكل على كينونة مشاركة هي علاقة «إرسالية» (SHIPMENT) تربط بين كينونة «عميل» (CUSTOMER) وكينونة «مورِّد» (VENDOR). كما يرتبط بالكينونة المشاركة ثلاث خصائص من ضِمنها الخاصية المعرِّفة «رقم الإرسالية» (Shipment_NO). وقد تمَّ تحديدُ هذه الخاصية معرفاً للكينونة المشاركة لسببين:

1- يُعَدُّ «رقم الإرسالية» مُعرِّفاً طبيعياً للكينونة المشاركة متعارفاً عليه في بيئة المستفيدين.

2- لا يمكن أن تُعرِّف خاصية «رمز العميل» المرتبطة بكينونة «عميل»، وخاصية «رمز المورد» المرتبطة بكينونة «مورد» حالات الكينونة المشاركة بشكلٍ منفرد؛ وذلك لأن المورِّد الواحد قد يُرسِل أكثر من إرسالية للعميل نفسه. وحتى لو تم استخدام بقية خصائص الكينونة المشاركة (وهي التاريخ والكمية)؛ إضافةً للخاصيتين السابقتين كمعرِّف للكينونة المشاركة؛ فإنه لا يمكِن التيقُّن من أن هذه الخصائص مجتمعة ستتمكَّن من تمييز حالات الكينونة المشاركة بشكلٍ منفرد. والسبب وراء ذلك؛ هو أن المورِّد الواحد قد يُرسِل لنفس العميل أكثر من إرسالية واحدة بنفس التاريخ وبنفس الكمية.

وباتباع خطوتَيْ التحويل أعلاه؛ يتمُّ إنشاء ثلاثة جداول، هي: جدول العميل (CUSTOMER) لتمثيل كينونة «عميل»، وجدول «المُورِّد» (VENDOR) لتمثيل كينونة «مُورِّد»، وجدول الإرسالية (SHIPMENT) لتمثيل العلاقة المشاركة «إرسالية». ونظراً لارتباط الكينونة المشاركة بخاصية معرفة وهي «رقم الإرسالية» (Shipment_NO)؛ فإنه يتمُّ تعريف خاصية المعرِّف بأنها المفتاح الرئيسي لجدول الكينونة المشاركة. كما يتمُّ تعريف المفتاح الرئيسي لجدول كينونة «عميل» وهو «رمز العميل» (Customer_ID) والمفتاح الرئيسي لجدول كينونة «مورِّد»، وهو «رمز المورِّد» (Vendor_ID) كحقول ضِمن جدول الإرسالية. ويتمُّ تعريف كلٍّ منهما على أنه مفتاح خارجي. كما يتمُّ تعريف بقية خصائص العلاقة المشاركة، وهي «التاريخ» (Date) و«الكمية» (Amount) بأنها حقولٌ ضِمن جدول الكينونة المشاركة. وقد تمَّ إيضاح المفتاح الرئيسي لجدول الإرسالية من خلال وَضْع خط متصل تحته، وإيضاح المفاتيح

الخارجية من خلال وَضْع خط متقطع تحت كل منهما. كما تمَّ إيضاح الجدول الذي يشيرُ إليه كل مفتاح خارجي من خلال السهم الذي يصل بين المفتاح والجدول الذي يشيرُ إليه المفتاح.

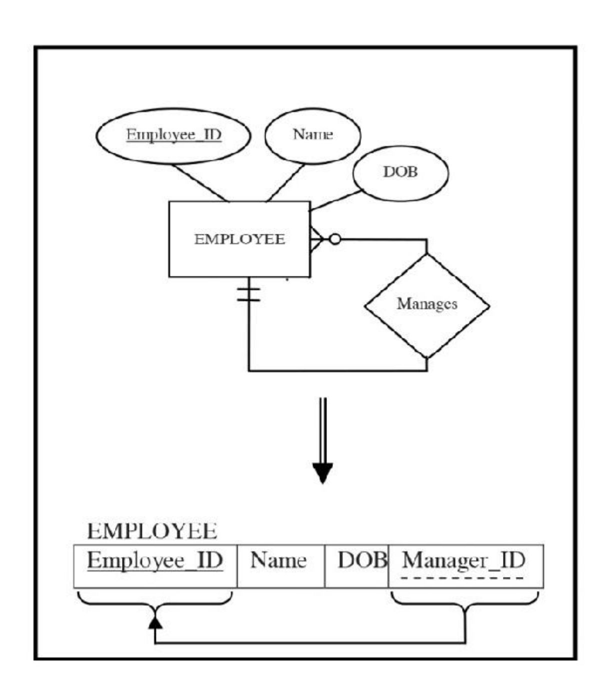
5–1–5 قاعدة التحويل الخامسة: التعامل مع العلاقات الأحادية:

العلاقة الأحادية؛ هي علاقةٌ تربط بين حالات الكينونة نفسها. وتُسمَّى هذه العلاقات في بعض الأحيان بالعلاقات المتواترة (ReCursive Relationship). ومن أهمِّ الحالات التي تظهر فيها العلاقات الأحادية هي عندما تكون تعدُّديتها «واحد - متعدد» و«متعدد - متعدد». وفيما يلي شرحٌ مُفصَّل لعملية تحويل كلٍّ منهما للنموذج العلاقي.

5–1–5–1 التعامل مع العلاقات الأحادية ذات التعدُّدية «واحد – متعدد»:

يتمُّ تحويل العلاقة الأحادية ذات التعدُّدية «واحد - متعدد» بخطوتين: في الخطوة الأولى يتمُّ تعريف جدول للكينونة التي تربط بين حالاتها العلاقة الأحادية كما سبق أن أوضحنا في قاعدة التحويل رقم (1) أعلاه. أمَّا في الخطوة الثانية؛ فيتمُّ إضافة حقل إضافي لجدول الكينونة الذي تمَّ تعريفه؛ بحيث يكون هذا الحقل مفتاحاً خارجياً يشيرُ إلى الجدول نفسه، وبحيث يكون مجاله (Domain) من نفس مجال المفتاح الرئيسي لجدول الكينونة الذي تمَّ تعريفه.

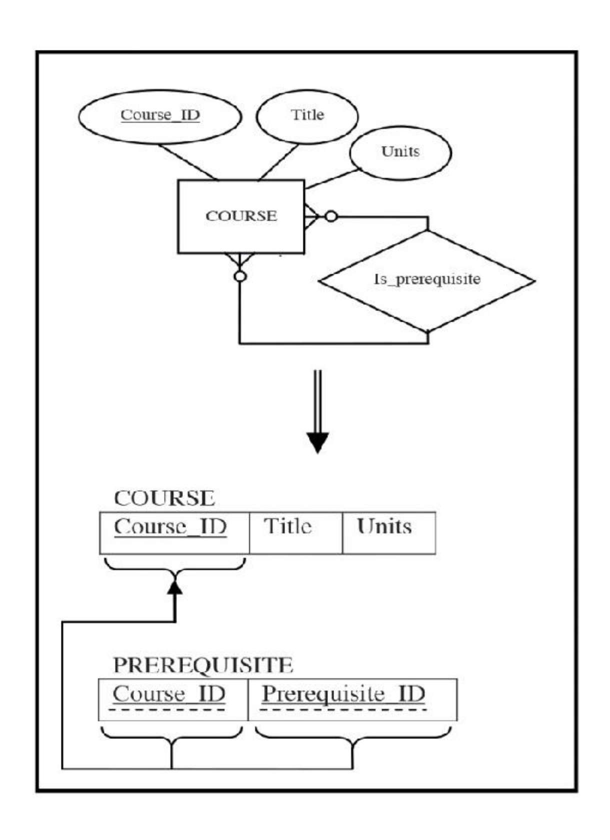
ويمثل الشكل رقم (5-11) علاقةً أحاديةً ذات تعدُّدية «واحد - متعدد» وهي علاقة «يدير» (Manages) التي تربط بين حالات الكينونة «موظف» (Employee). فالمدير الواحد في المنظمة يديرُ صفراً أو أكثر من الموظفين. أمَّا الموظف الواحد؛ فيجب أن يرأسه (أو يديره) مديرٌ واحد فقط[2]. وحسب قاعدة التحويل أعلاه؛ يتمُّ تعريف جدول لكينونة «موظف»؛ بحيث يحتوي على الخصائص المرتبطة بالكينونة، وهي: رمز الموظف، واسم الموظف، وتاريخ ميلاده. كما يتمُّ تعريف المفتاح الرئيسي للجدول وهو رمز الموظف. بعد ذلك تتمُّ إضافة حقلٍ جديد للجدول، وهو حقل «رمز المدير»؛ وذلك لتمثيل علاقة «يدير». كما يتمُّ تعريف الحقل الجديد؛ بحيث يكون من نفس مجال المفتاح الرئيسي لجدول كينونة «موظف»، وعلى أساس أنه مفتاح خارجي يشير إلى جدول الكينونة نفسه. وبهذه الطريقة يُمكِن التعرُّف على مدير كلِّ موظف في المنظمة من خلال المفتاح الخارجي المدوَّن في سجل الموظف.



شكل رقم (5–11): تحويل العلاقة الأحادية ذات التعدُّدية «واحد – متعدد» إلى النموذج العلاقي.

2-5-1-5 التعامل مع العلاقات الأحادية ذات التعدُّدية «متعدد - متعدد»:

عند وجود علاقة أحادية ذات تعدُّدية «متعدد - متعدد»؛ فإنه يتمُّ تعريف جدولين في أثناء عملية التحويل للنموذج العلاقي. الجدول الأول يمثل الكينونة التي ترتبط بالعلاقة والجدول الثاني يمثل العلاقة نفسها. ويكون المفتاح الرئيسي للجدول الذي يمثل العلاقة عبارة عن مفتاح مُركَّب يُعرَّف فيه المفتاح الرئيسي لجدول الكينونة مرتين، كما يُعرَّف كلُّ جزءٍ منه على أنه مفتاح خارجي يشيرُ إلى جدول الكينونة. ويمكِن تشبيه عملية التحويل هذه بعملية تحويل العلاقة الثنائية ذات التعددية «متعدد - متعدد» التي يتمُّ فيها تعريف ثلاثة جداول؛ بحيث يكون إحدى هذه الجداول ممثلاً للعلاقة التي تربط بين الكينونتين المرتبطتين بها، ويتكوَّن مفتاحها الرئيسي من حقول المفاتيح الرئيسية لكلتا الكينونتين التي تربط بينهما. ونظراً لوجود كينونة واحدة في أيَّة علاقة أحادية؛ فإن المفتاح الرئيسي لجدول العلاقة هو تكرار المفتاح الرئيسي لجدول الكينونة التي ترتبط بالعلاقة. وفي حال ارتبطت العلاقة بخصائص؛ فإنه يتمُّ تعريف حقل مقابل لكلِّ خاصية ضمن جدول العلاقة.



شكل رقم (5–12): تحويل العلاقة الأحادية ذات التعدُّدية «متعدد – متعدد» إلى النموذج العلاقي.

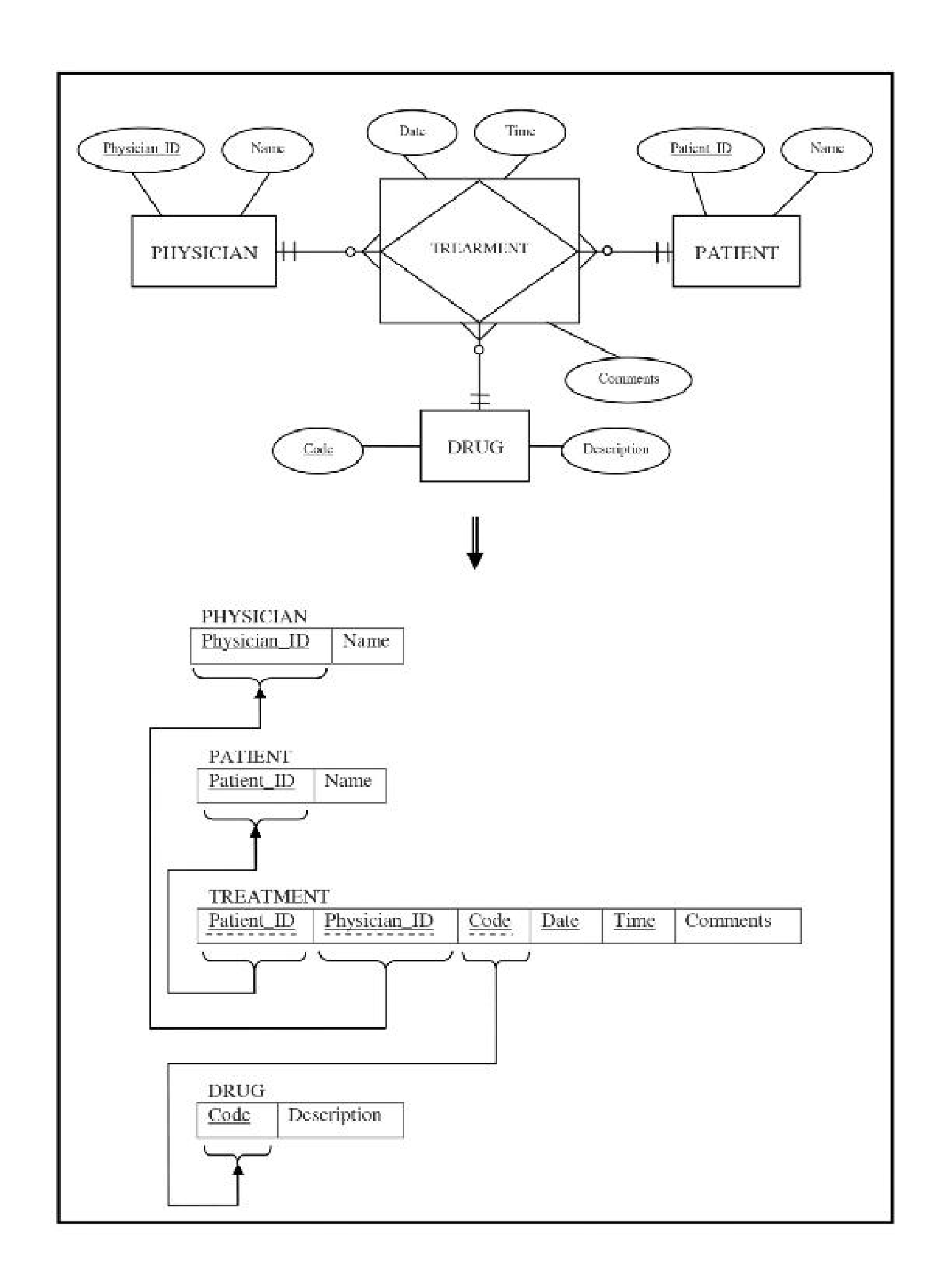
ويُوضِّح الشكل رقم (5-12) عملية تحويل علاقة أحادية ذات تعدُّدية «متعدد ـ متعدد». ويُوجَد في الشكل كينونة «المادة الدراسية» (COURSE) التي ترتبط حالاتها مع بعض من خلال علاقة «متطلب دراسي» (Is_prerequisite). وتعني هذه العلاقة؛ أن كل مادة دراسية لها صفر أو أكثر من المتطلبات الدراسية. كما أن المادة الدراسية قد تكون متطلباً دراسياً لصفر أو أكثر من المواد الدراسية. وحسب قاعدة التحويل أعلاه؛ يتمُّ تعريف جدولين: أحدهما لتمثيل كينونة «المادة الدراسية» والآخر لتمثيل علاقة «متطلب دراسي». ونظراً لأن علاقة «متطلب دراسي» لا ترتبط بأية خصائص وأن المفتاح الرئيسي لجدول المادة الدراسية يتكوَّن من حقل واحد فقط؛ فإن جدول العلاقة يحتوي على حقلين فقط. وكل حقل منهما هو المفتاح الرئيسي لجدول الكينونة، مع الملاحظة بأنَّ تسمية حقل المفتاح الرئيسي لجدول الكينونة ليس من الضروري أن يكون نفسه في الجدول الآخر ما دام من نفس المجال. ويُصبح المفتاح الرئيسي لجدول العلاقة هو نسختي حقل المفتاح الرئيسي لجدول الكينونة مُدمَجين مع بعضهما. كما يتمُّ تعريف كلٍّ واحد منهما على أنه مفتاح خارجي يشيرُ لجدول الكينونة، كما هو مُوضَّح في الشكل رقم (5-12).

5–1–6 قاعدة التحويل السادسة: التعامل مع العلاقات الثلاثية (وما أعلى من ذلك):

العلاقة الثلاثية؛ هي علاقةٌ تربط بـين ثلاثة أنواع من الكينونات. ويُفضَّل تحويل العلاقة الثلاثية (والعلاقات ذات الدرجات الأعلى من ثلاثة) إلى علاقة مشاركة؛ حتى يمكن توصيف قيود التعدُّدية بـشكلٍ أدق. ولتحويل علاقة مشاركة تربط بين ثلاث كينونات (أو أكثر)، يتمُّ إنشاء جدول لتمثيل العلاقة المشاركة (بالإضافة إلى جداول الكينونات التي تربط بينها العلاقة)؛ بحيث يحتوي الجدول على حقول تمثل المفاتيح الرئيسية لجداول الكينونات التي تربط بينها العلاقة وحقول لتمثيل أيَّة خصائص مرتبطة بـالعلاقة نفسها. ويكون المفتاح الرئيسي لجدول العلاقة مكوناً من حقول المفاتيح الرئيسية لجداول الكينونات التي تربط بـينها العلاقة. وفي بعض الأحيان يُضاف لحقول المفاتيح الرئيسية لجداول الكينونات التي تربطها العلاقة حقول أخرى تمثل بعض خصائص العلاقة نفسها؛ وذلك عندما لا تمكِّننا المفاتيح الرئيسية لجداول الكينونات من التعرُّف على حالات العلاقة بـشكلٍ متفرد.

ويُمثِّل الشكل رقم (5-13) علاقة ثلاثية وهي علاقة «علاج» (TREATMENT) التي تربط بين كينونة «طبيب» (PHYSICIAN) وكينونة «مريض» (PATIENT) وكينونة «دواء» (DRUG). ويُمكِن أن تُقرَأ هذه العلاقة على أن الطبيب يصف دواءً للمريض، وهو علاج المريض. وعليه؛ فإن علاقة «علاج» هي علاقة تربط بين ثلاث حالات في الوقت نفسه: حالة من حالات الأطباء، وحالة من حالات المرضى، وحالة من حالات الدواء.

وحسب قاعدة التحويل أعلاه؛ يتمُّ إنشاء جدول خاص بالعلاقة الثلاثية (أو العلاقة ذات الدرجة الأعلى من ثلاث)، بالإضافة لجداول الكينونات التي تربط بـينها العلاقة. ويحتوي جدول العلاقة على حقولٍ تمثِّل المفاتيح الرئيسية لجداول الكينونات الثلاث، وهي: «رمز الطبيب» (PhysiCian_ID)، و«رمز المريض» (Patient_ID)، و«رمز الدواء» (Code). وتكون هذه الحقول الثلاثة جزءاً من المفتاح الرئيسي لجدول العلاقة؛ إضافةً إلى كونها مفاتيح خارجية تشير لجداول الكينونات الثلاث التي تربط بـينها العلاقة. كما يتكون جدول العلاقة من حقول الخصائص المرتبطة بالعلاقة نفسها، وهي: «التاريخ» (Date)، و«الوقت» (Time)، و«الملاحظات» (Comments).



شكل رقم (5–13): تحويل العلاقة الثلاثية إلى النموذج العلاقي.

ونظراً لأنه من الممكن أن يقوم المريض الواحد بمقابلة الطبيب نفسه وأخذ الدواء نفسه أكثر من مرة في اليوم الواحد؛ فإن المفاتيح الرئيسية لجداول الكينونات الثلاث فقط لا تصلح لأن تكون مفتاحاً رئيسياً لجدول العلاقة؛ وذلك لأنها لا تمكِّن من التمييز بين حالات العلاقة بـشكلٍ منفرد. لذلك تمَّ استخدام خاصية التاريخ وخاصية الوقت المرتبطتين بـالعلاقة؛ ليكونا جزءاً من المفتاح الرئيسي لجدول العلاقة. وبهذه الطريقة يمكِن التعرُّف على حالات العلاقة بشكلٍ منفرد؛ إذ إن المريض قد يقوم بمقابلة الطبيب نفسه وأخذ الدواء نفسه في اليوم نفسه ولكن في أوقات مختلفة. أما

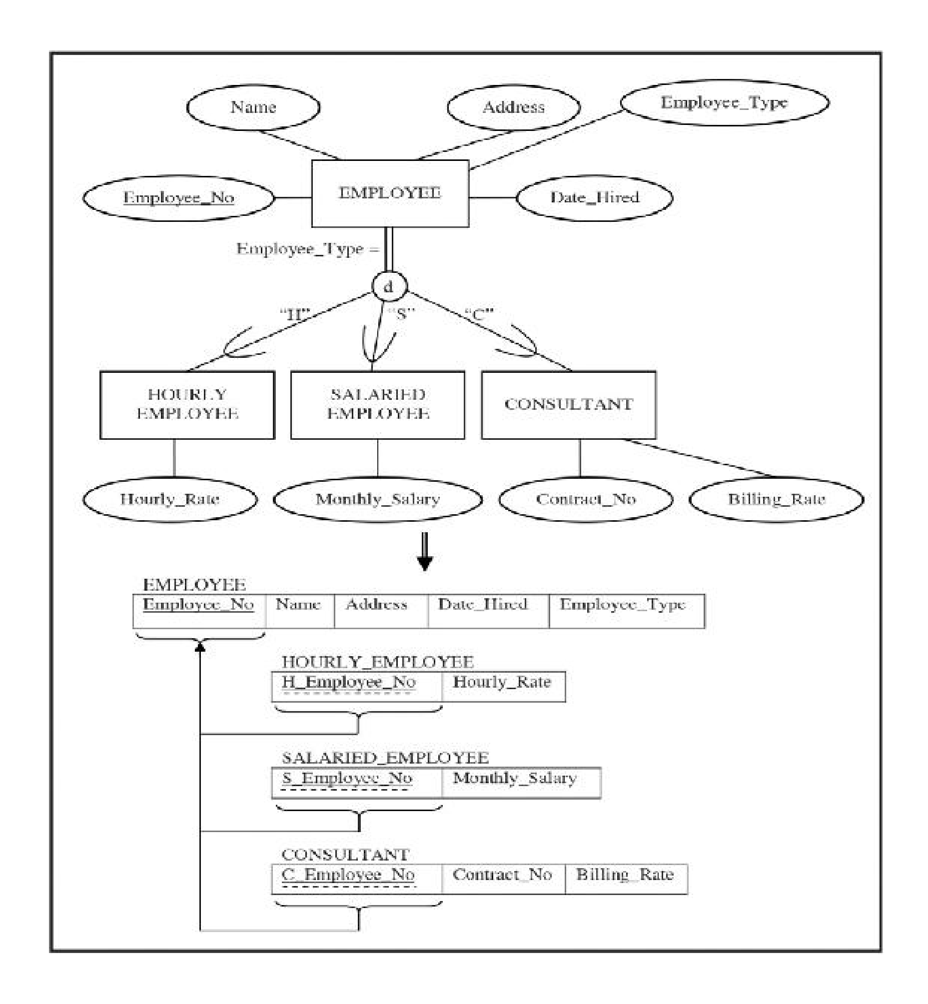
إذا افترضنا أنه ليس من الممكن أن يقوم المريض بمقابلة الطبيب نفسه وأخذ الدواء نفسه في اليوم نفسه؛ فإنه يمكن الاكتفاء بخاصية التاريخ لتصبح جزءاً من المفتاح الرئيسي دون استخدام خاصية الوقت.

5–1–7 قاعدة التحويل السابعة: التعامل مع علاقات الأنواع الرئيسية والأنواع الفرعية:

لا يُمكِّن النموذج العلاقي حالياً من تمثيل علاقات الأنواع الرئيسية والأنواع الفرعية بشكلٍ مباشر؛ غير أنه يتوفر عددٌ من الخيارات لمصمِّمي قواعد البيانات تمكِّنهم من تحويل علاقات الأنواع الرئيسية والأنواع الفرعية من النموذج المفاهيمي «كينونة ـ علاقة» إلى النموذج العلاقي. وفيما يلي شرحٌ للخيارات الأربعة التي تُعَدُّ الأكثر شيوعاً في عملية التحويل (Elmasri and Navathe, 2015):

5–1–7–1 الخيار الأول:

يتمُّ إنشاء عددٍ من الجداول؛ بحيث يُخصَّص واحدٌ منها لتمثيل النوع الرئيسي، وواحدٌ لكلِّ نوعٍ من أنواعه الفرعية. ويتكوَّن جدول النوع الرئيسي من عددٍ من الحقول يكون مكافئاً لعددِ الخصائص المرتبطة به في نموذج «كينونة ـ علاقة»، ويكون المفتاح الرئيسي لجدول النوع الرئيسي هو الخاصية (أو مجموعة الخصائص) المعرِّفة للنوع الرئيسي في نموذج «كينونة ـ علاقة». كما تتمُّ إضافة حقل أو أكثر في جدول النوع الرئيسي لتمثيل «مميِّز الأنواع الفرعية». ويتمُّ إنشاء جدولٍ لكلِّ نوعٍ فرعي يرتبط بالنوع الرئيسي في نموذج «كينونة – علاقة»؛ بحيث يكون عددُ حقول الجدول المنشأ لنوع فرعي معين مكافئاً لعدد خصائص النوع الفرعي في نموذج «كينونة ـ علاقة»، بالإضافة إلى حقل (أو أكثر) لتمثيل المفتاح الرئيسي لجدول النوع الرئيسي. ويعني هذا أن كلَّ جدول لنوع فرعي يجب أن يحتوي على المفتاح الرئيسي لجدول النوع الرئيسي. كما يتمُّ تعريف حقل المفتاح الرئيسي للنوع الرئيسي الذي تمَّ إنشاؤه في جدول النوع الفرعي على أنه مفتاح رئيسي للنوع الفرعي، وفي الوقت نفسه مفتاح خارجي يشير إلى جدول النوع الرئيسي.



شكل رقم (5–14): تحويل علاقة الأنواع الرئيسية والأنواع الفرعية إلى النموذج العلاقي وفق الخيار الأول للتحويل.

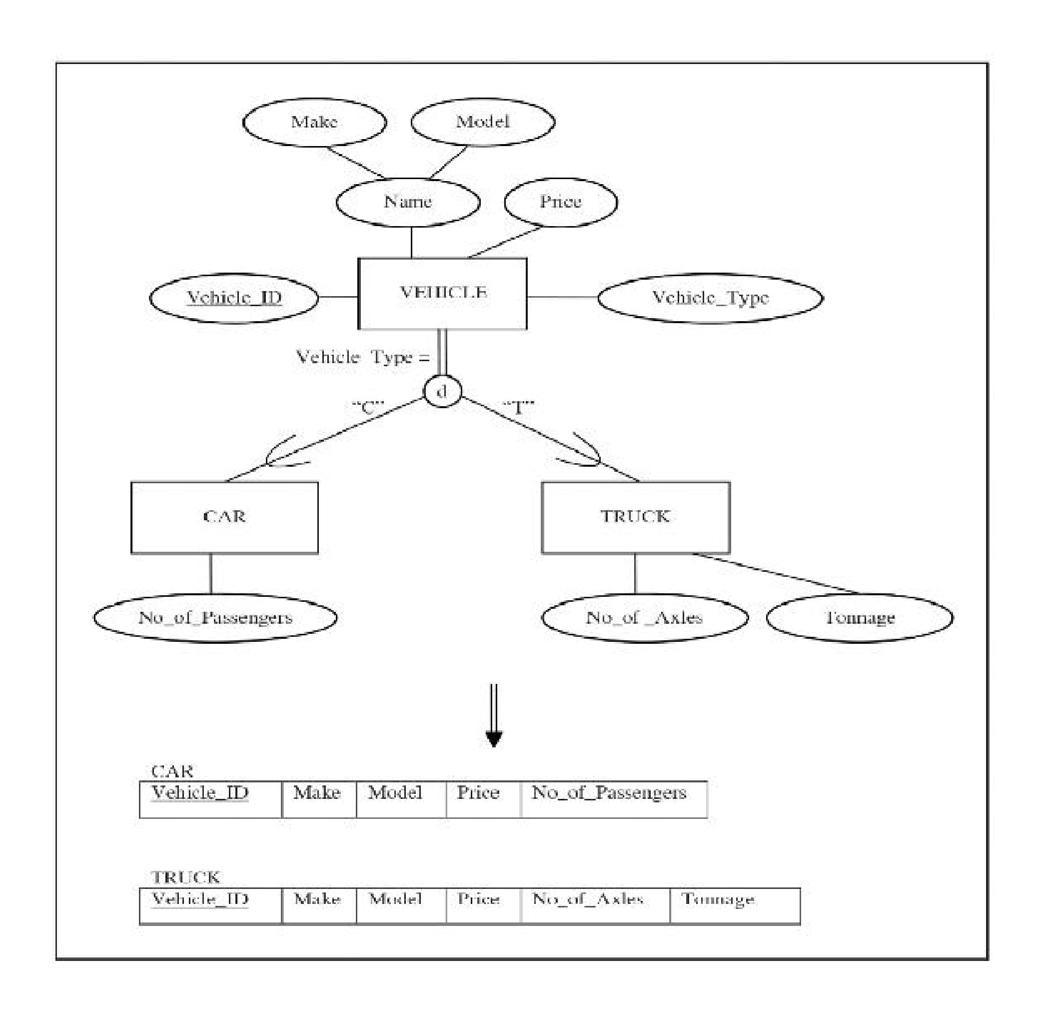
ويمثِّل الشكل رقم (5-14) نوعاً رئيسياً، وهو «موظف» (EMPLOYEE) يرتبط به ثلاثة أنواع فرعية من الموظفين، وهي: «موظفو أجر الساعات» (HOURLY_EMPLOYEE)، و«موظفو الأجر الشهري» (SALARIED_EMPOYEE)، و«المستشارون» (CONSULTANTS). وحسب القاعدة أعلاه؛ يتمُّ إنشاء أربعة جداول، أحدها لتمثيل النوع الرئيسي والثلاثة المتبقية لتمثيل الأنواع الفرعية. كما يتمُّ إدراج الخصائص المشتركة للأنواع الفرعية (وهي تلك المرتبطة بالنوع الرئيسي) كحقول ضِمن جدول النوع الرئيسي بما فيها الخاصية المعرَّفة، وهي خاصية «رقم الموظف» (Employee_No). يتمُّ أيضاً إضافة حقل للخاصية التي تميِّز بين الأنواع الفرعية، وهي خاصية «نوع الموظف» (Employee_Type). أمَّا بالنسبة لجداول الأنواع الفرعية؛

فيتكون كلٌّ واحدٍ منها من حقول تمثل الخصائص التي يتفرَّد بها عن بقية الأنواع الفرعية؛ إضافةً إلى حقل المفتاح الرئيسي لجدول النوع الرئيسي الذي يُعرف ضِمن جدول النوع الفرعي على أساس أنه مفتاح رئيسي، وفي الوقت نفسه مفتاح خارجي يشير إلى جدول النوع الرئيسي.

5–1–7–2 الخيار الثاني:

يتمُّ إنشاء عددٍ من الجداول؛ بحيث يكون كلُّ واحدٍ منها ممثلاً لنوع فرعي واحد دون تمثيل النوع الرئيسي. ويحتوي جدول كلِّ نوعٍ فرعي على حقول لتمثيل الخصائص التي ينفرد فيها النوع الفرعي؛ بالإضافة إلى الخصائص المشتركة بين الأنواع الفرعية؛ وهي تلك المرتبطة بالنوع الرئيسي. ويكون المفتاح الرئيسي لجدول أيِّ نوعٍ فرعي هو الحقل الذي يمثل الخاصية المعرفة التي ترتبط بكينونة النوع الرئيسي. كما يُستغنَى عن تمثيل خاصية «مميز الأنواع الفرعية» باستخدام طريقة التحويل هذه. ويُمكِن استخدام طريقة التحويل هذه عندما يكون قيد التخصيص كاملاً بمعنى أن أيَّة حالة من حالات النوع الرئيسي لا بد أن تُوجَد ضِمن أنواعه الفرعية، ويكون قيد الانفصال كاملاً أيضاً، بمعنى أنه لا يمكن أن توجد حالة ما ضمن أكثر من نوعٍ فرعي واحد في وقتٍ واحد.

ويوضِّح الشكل رقم (5-15) كينونة النوع الرئيسي، وهو «مركبة» التي يرتبط فيها نوعان فرعيان هما: «سيارة» و«شاحنة». ونظراً لأن كلَّ مركبة لا بد أن تمثل ضِمن الأنواع الفرعية؛ فإن قيد التخصيص هو تخصيصٌ كاملٌ، كما هو مُوضَّح في الشكل بالخطين المزدوجين اللذين يصلان النوع الرئيسي بنقطة التفرُّع. أما قيد الانفصال؛ فهو انفصالٌ كاملٌ؛ وذلك لأن السيارة لا يمكن أن تكون شاحنة أو بالعكس. في مثل هذه الحالة يمكن استخدام الطريقة أعلاه في عملية تحويل النموذج المفاهيمي إلى النموذج العلاقي؛ إذ يتمُّ إنشاء جدولين أحدهما لتمثيل النوع الفرعي «سيارة»، والثاني لتمثيل النوع الفرعي «شاحنة». كما يتمُّ إدراج حقول لتمثيل الخصائص المشتركة (التي ترتبط بالنوع الرئيسي) ضِمن جدولي كلا النوعين الفرعيين بالإضافة للخصائص المميزة لكلِّ نوعٍ منهما ضمن الجدول الممثل للنوع الفرعي. كما تعرف خاصية المميز المرتبطة بالنوع الرئيسي، في كلا الجدولين، على أنها المفتاح الرئيسي لكلٍّ منهما.



CAR

Vehicle_ID	Make	Model	Price	No_of_Passengers

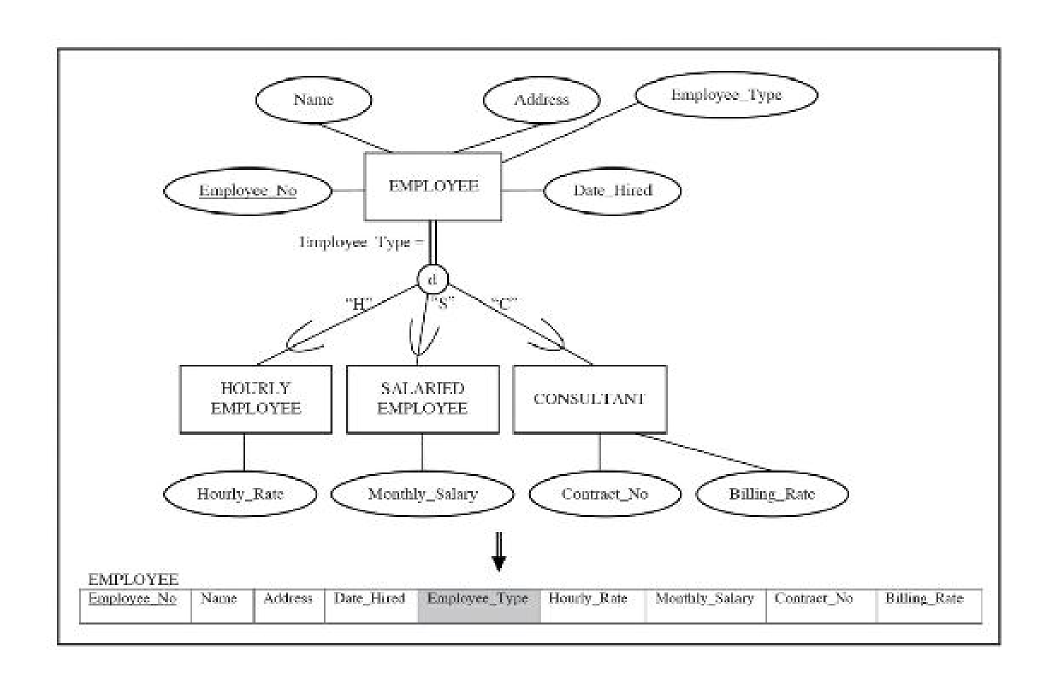
TRUCK

Vehicle_ID	Make	Model	Price	No_of_Axles	Tonnage

شكل رقم (5–15): تحويل علاقة الأنواع الرئيسية والأنواع الفرعية إلى النموذج العلاقي وفق الخيار الثاني للتحويل.

5–1–7–3 الخيار الثالث:

يتمُّ إنشاءُ جدولٍ واحد فقط يتكون من حقول تمثل جميع الخصائص المشتركة للأنواع الفرعية؛ بالإضافة إلى حقول تمثِّل الخصائص المرتبطة بـكلِّ نوع فرعي. كما يتمُّ إضافة حقل لتمثيل خاصية «مميز الأنواع الفرعية». ويمكن استخدام طريقة التحويل هذه عندما يكون قيد الانفصال كاملاً، بمعنى أنه لا يمكن أن تُوجَد حالة ما ضِمن أكثر من نوع فرعي واحد في الوقت نفسه، بغض النظر عن قيد التخصيص، سواء أكان كاملاً أو جزئياً.



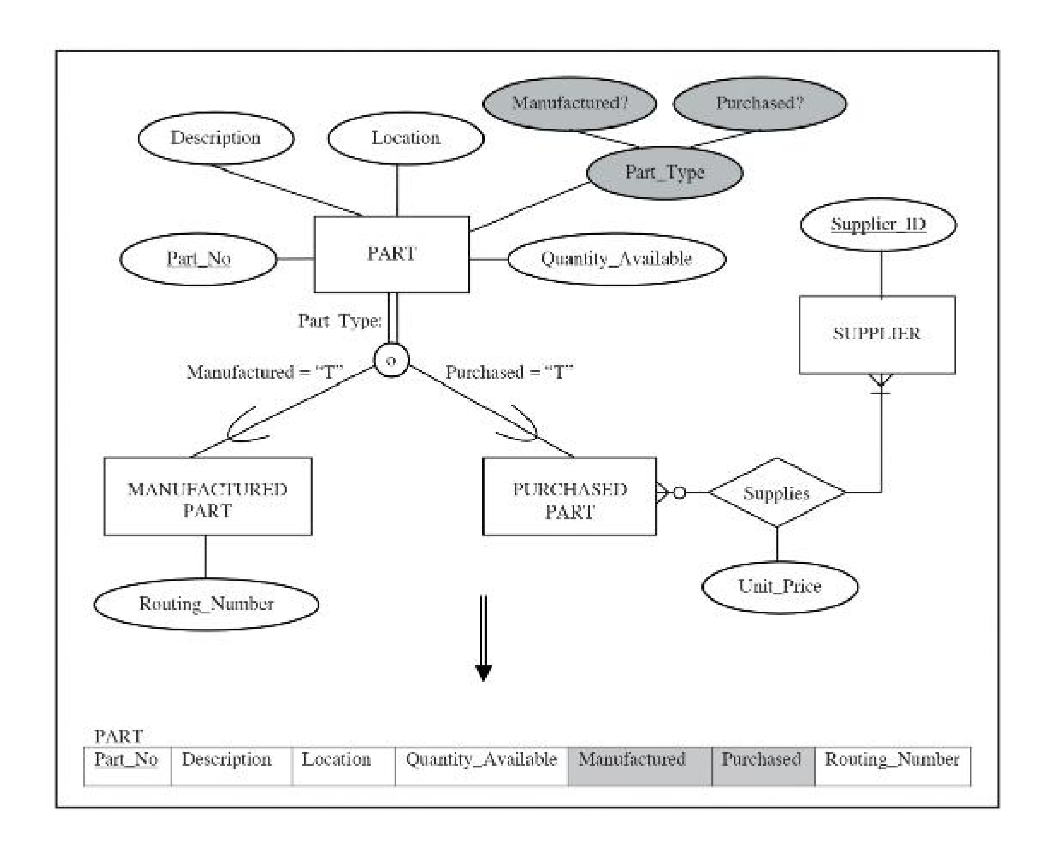
شكل رقم (5–16): تحويل علاقة الأنواع الرئيسية والأنواع الفرعية إلى النموذج العلاقي وفق الخيار الثالث للتحويل.

ويمثل الشكل رقم (5-16) نوعاً رئيسياً وهو «موظف» (EMPLOYEE) الذي يرتبط به ثلاثة أنواع فرعية من الموظفين، وهي: «موظفو أجر الساعات» (HOURLY_EMPLOYEE)، و«موظفو الأجر الشهري» (SALARIED_EMPLOYEE)، و«المستشارون» (CONSULTANTS). ونظراً لكون قيد الانفصال هو انفصال كامل؛ فإنه يمكن تطبيق القاعدة أعلاه؛ بحيث يتمُّ إنشاء جدول واحد فقط لتمثيل النوع الرئيسي والأنواع الفرعية الثلاثة. ويتكون الجدول من حقول تمثل الخصائص المشتركة للأنواع الفرعية (وهي تلك المرتبطة بالنوع الرئيسي)؛ بالإضافة إلى حقول تمثل الخصائص المرتبطة بكلِّ نوع فرعي. كما يحتوي الجدول على حقل يمثل مميز الأنواع الفرعية، وهو حقل «نوع الموظف» (Emplyee_Type). وعند إضافة موظف إلى الجدول؛ يتمُّ إدخال كافة بيانات الموظف ونوعه. أما بالنسبة للحقول التي لا تنطبق على الموظف فتكون قيمها غائبةً (NULL).

5–1–7–4 الخيار الرابع:

يتمُّ إنشاء جدول واحد فقط يتكون من حقولٍ تمثل جميع الخصائص المشتركة للأنواع الفرعية؛ إضافةً إلى حقول تمثل الخصائص المرتبطة بكلِّ نوع فرعي. كما يتمُّ إضافة عددٍ من الحقول يساوي عدد الأنواع الفرعية؛ بحيث يقابل كلُّ حقل منها نوعاً فرعياً واحداً. وتُعرَّف هذه الحقول الإضافية بأنها ذات بيانات «ثنائية القيم» (Boolean Data Type). وتكون قيمة أيِّ حقل من هذه الحقول الإضافية مساويةً للقيمة «صح» (True) إذا كانت الحالة المُدخلة تتبع للنوع الفرعي المقابل للحقل الإضافي ثنائي القيم. أمَّا إذا لم تكن الحالة المُدخَلة تابعةً لذات النوع الفرعي، تكون قيمة الحقل الإضافي المقابل للنوع الفرعي هي «خطأ» (False). ويُمكِن استخدام طريقة التحويل هذه عندما يكون قيد الانفصال جزئياً بمعنى أنه من الممكن أن تُوجَد حالة ما ضِمن أكثر من نوع فرعي واحد في الوقت نفسه، بغض النظر عن قيد التخصيص، سواء كان كاملاً أم جزئياً. كما يُمكِن استخدام هذه الطريقة أيضاً عندما يكون قيد الانفصال كاملاً.

ويمثل الشكل رقم (5-17) نوعاً رئيسياً وهو «قطعة غيار» (PART) يرتبط به نوعان فرعيان، هما: «قطع الغيار المصنَّعة داخلياً» (في المنظمة نفسها) بمسمَّى (MANUFACTURED_PART)، و «قطع الغيار المشتراة» (PURCHASED_PART). ونظراً لكون قيد الانفصال هو انفصال متداخل؛ إذ إن بعض قطع الغيار قد تكون مُصنَّعة داخلياً وفي الوقت نفسه مُشتراة؛ يمكن حينئذٍ تطبيق القاعدة أعلاه؛ بحيث يتمُّ إنشاء جدول واحد فقط لتمثيل النوع الرئيسي ونوعيه الفرعين. ويتكون الجدول من حقول تمثل الخصائص المشتركة للأنواع الفرعية (وهي تلك المرتبطة بالنوع الرئيسي)؛ إضافةً إلى حقول تمثل الخصائص المرتبطة بكلِّ نوع فرعي. كما يحتوي الجدول على حقلين ذَوَيْ نوعية بيانات ثنائية القيم، هما: حقل «مُصنَّعة» (ManufaCtured) وحقل «مُشتراة» (PurChased). وتكون قيمة أيِّ حقل من هذين الحقلين إما «صح» وإما «خطأ». فعندما تكون قطعة الغيار مُشتراة فقط تكون قيمة حقل «مُشتراة» صح، وتكون قيمة حقل «مُصنَّعة» خطأ. أمَّا إذا كانت قطعة الغيار مُصنَّعة داخلياً فقط؛ فتكون قيمة حقل «مُصنَّعة» صح، وقيمة حقل «مُشتراة» خطأ. وفي حال كانت بعض من قطع الغيار مُصنَّعة داخلياً وبعض منها مشتراة؛ فإن قيمة كلا الحقلين تكون صحيحةً.



شكل رقم (5–17): تحويل علاقة الأنواع الرئيسية، والأنواع الفرعية إلى النموذج العلاقي وفق الخيار الرابع للتحويل.

5–1–7–5 فوارق خيارات تصميم علاقات الأنواع الرئيسية والأنواع الفرعية:

يُعَدُّ الخيار الأول والخيار الثاني من خيارات التصميم التي ينتج عنها أكثر من جدول، في حين أن الخيار الثالث والخيار الرابع يُعدَّان من خيارات التصميم التي ينتج عنها جدول واحد فقط. كما يُعَدُّ الخيار الأول لعملية تحويل علاقات الأنواع الرئيسية والأنواع الفرعية خياراً عاماً، بمعنى أنه يمكن استخدامه بغض النظر عن القيود المفروضة على النوع الرئيسي وأنواعه الفرعية (وهما قيد التخصيص وقيد الانفصال). إلا أن هذا الخيار يتطلب عملية «ربط تساوي» (Equi-Join) على المفتاح الرئيسي بين جدول النوع الرئيسي وجدول أيِّ نوع فرعي للحصول على جميع بيانات النوع الفرعي. أمَّا إذا أردنا الحصول على بيانات جميع الأنواع الفرعية؛ فإن هذا يتطلب عملية «اتحاد خارجي» (Outer Union) بعد عملية «ربط تساوي» بين الأنواع الفرعية بالنوع الرئيسي.

أمَّا الخيار الثاني فلا يتطلب عملية «ربط تساوي» للحصول على جميع بيانات نوع فرعي معين؛ لأن هذه البيانات متوفرة بالكامل ضِمن جدول النوع الفرعي نفسه. غير أن هذا الخيار يُستخدَم في حالة كون قيد التخصيص كاملاً، وقيد الانفصال كاملاً أيضاً. فإذا لم يكن قيد التخصيص كاملاً؛ فإنه سيتمُّ فقد بيانات الأنواع التي لم يتم تخصيصها. أما إذا كان قيد الانفصال متداخلاً؛ فإنه سيتمُّ تكرار تخزين بعض البيانات ضِمن جداول الأنواع الفرعية. وكما هو الحال في الخيار الأول، تُستخدَم عملية «الاتحاد الخارجي» (Outer Union) إذا أردنا الحصول على جميع الأنواع الفرعية؛ وذلك لأنه لا يحتوي أيٌّ من جداول الأنواع الفرعية على بيانات الأنواع الفرعية كافةً.

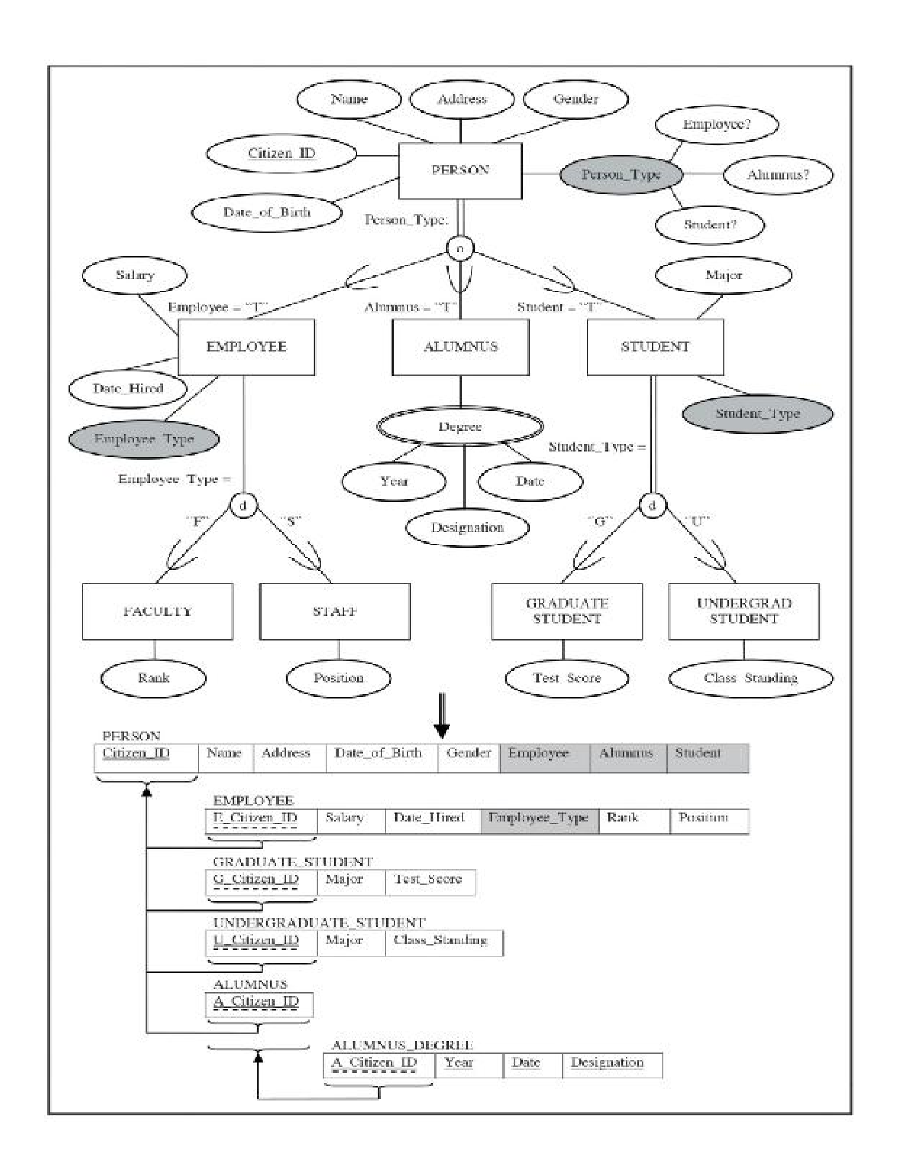
يُستخدَم الخيار الثالث عندما يكون قيد الانفصال كاملاً. ويتمُّ استخدام أحد حقول الجدول باعتباره مميزاً للنوع الفرعي؛ بحيث تُحدِّد القيمة المخزَّنة في هذا الحقل لكلِّ حالة مُخزَّنة في الجدول؛ النوعَ الفرعي الذي تتبعه الحالة. وعندما يكون قيد التخصيص كاملاً؛ فإنه لا بد أن تتبع كلُّ حالة مخزَّنة في الجدول لأحد الأنواع الفرعية؛ مما يعني وجود قيمة في حقل مميز الأنواع الفرعية تحَدِّد النوع الفرعي الذي تتبعه الحالة. أما إذا كان قيد التخصيص جزئياً؛ فإن هذا يعني إمكانية وجود حالات ضِمن الجدول لا تتبع لأيِّ نوع فرعي. وفي هذه الحالة تترك قيمة حقل مميز الأنواع الفرعية غائِبةً (NULL) كما تترك جميع الحقول التابعة للأنواع الفرعية غائبة أيضاً.

صُمِّمَ الخيارُ الرابع بحيث يُمكِّن من تمثيل علاقة الأنواع الرئيسية والأنواع الفرعية عندما يكون قيد الانفصال متداخلاً. وباستخدام هذا الخيار يتمُّ تعريف عددٍ إضافي من الحقول في الجدول الذي يمثل النوع الرئيسي والأنواع الفرعية؛ بحيث يكون عدد الحقول الإضافية هذه مساوياً لعدد الأنواع الفرعية. وتُعرف هذه الحقول الإضافية على أنها ذات بيانات «ثنائية القيم» (Boolean Data Type). وتكون قِيم أيِّ حقل من هذه الحقول الإضافية مساويةً للقيمة «صح» (True) إذا كانت الحالة المُدخلَة تتبع للنوع الفرعي المقابل للحقل الإضافي ثنائي القيم. أما إذا لم تكن الحالة المُدخَلة تابعةً للنوع الفرعي نفسه؛ تكون قيمة الحقل الإضافي المقابل للنوع الفرعي هي «خطأ» (False). كما تكون قِيم جميع الحقول التابعة للنوع الفرعي، في هذه الحالة، غائبةً (NULL). وكما هو الحال في طريقة التحويل الثالثة؛ فإن طريقة التحويل هذه تعفينا من إجراء أية عملية ربط أو اتحاد للحصول على كامل بيانات الأنواع الفرعية.

إن الخيارات الأربعة أعلاه تعطي مُصمِّمي قواعد البيانات المرونة الكافية لتحديد الطريقة المناسبة في تحويل علاقات الأنواع الرئيسية والأنواع الفرعية. فالخيار الأول والخيار الثاني ينتج عنهما أكثر من جدول؛ مما يستدعي إجراء عمليات ربط بين الجدول واتحاد فيما بينها؛ الأمر الذي يتطلب استغراق وقتٍ أطول في تنفيذ الاستفسارات مقارنةً بالطريقة الثالثة والطريقة الرابعة. غير أن الطريقة الأولى والطريقة الثانية لا تستنزفان المساحة التخزينية؛ وذلك لعدم وجود الكثير من القِيم الغائبة ضِمن الجداول مقارنةً بالطريقة الثالثة والطريقة الرابعة. على النقيض من ذلك؛ فإن الطريقة الثالثة والطريقة الرابعة أسرع في تنفيذ الاستفسارات من الطريقة الأولى والطريقة الثانية؛ وذلك لكون جميع بيانات الأنواع الفرعية متوافرة في جدولٍ واحد؛ مما يعني عدم الحاجة إلى إجراء أية عملية ربط أو اتحاد؛ غير أن هاتين الطريقتين تستنزفان الكثير من المساحة التخزينية، وخاصةً عندما تكون الحقول المرتبطة بكلِّ نوعٍ فرعي كثيرةً نسبياً؛ مما ينتج عنه الكثيرُ من القِيم الغائبة ضِمن بيانات الجدول.

5–1–7–6 تحويل هرميات الأنواع الرئيسية والأنواع الفرعية:

عندما نقوم بعملية تحويل هرميات من علاقات الأنواع الرئيسية والأنواع الفرعية؛ فإنه ليس من الضروري اتباع نفس خيار التحويل لجميع الأنواع الرئيسية والأنواع الفرعية، وإنما يمكن استخدام خيارات مختلفة. ويوضِّح الشكل رقم (5-18) خيارات مختلفة لتحويل هرمية من الأنواع الرئيسية والأنواع الفرعية.



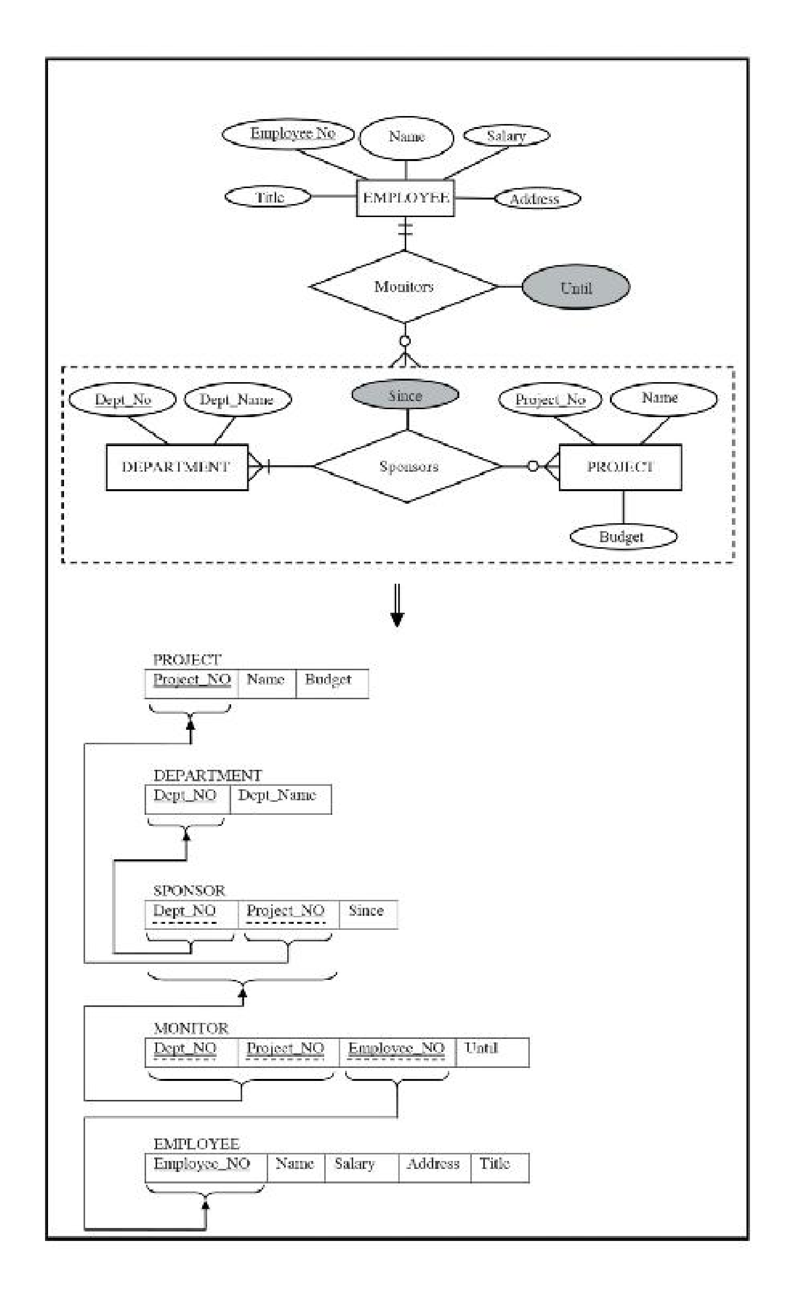
شكل رقم (5–18): تحويل هرميات الأنواع الرئيسية والأنواع الفرعية إلى النموذج العلاقي.

ويُلاحَظ في عملية التحويل المُمثَّلة في الشكل أنه تمَّ استخدامُ الخيار الأول لتمثيل علاقة النوع الرئيسي «شخص» (PERSON) وأنواعه الفرعية؛ إذ تمَّ إدراجُ الخصائص المشتركة لجميع فئات الأشخاص ضِمن جدول الكينونة الرئيسية «شخص». ولكون قيد الانفصال انفصالاً متداخلاً؛ فقد تمَّ إضافة ثلاثة حقول تبيِّن نوعية الشخص فيما إذا كان موظفاً أو طالباً أو خريجاً أو أية توليفات أخرى، مثل أن يكون الشخص خريجاً من الجامعة، وفي الوقت نفسه موظفاً فيها. وعند تحويل النوع الفرعي «موظف» تمَّ استخدام الخيار الثالث؛ إذ تمَّ إنشاء جدول واحد لجميع أنواع الموظفين مع إضافة مميِّز لنوع الموظف يبيِّن إنْ كان الموظف عضواً لهيئة التدريس أو موظفاً غير ذلك (من العاملين في إحدى الوظائف الإدارية). وعند تحويل النوع الفرعي «طالب»

(STUDENT) تمَّ استخدامُ الخيار الثاني؛ إذ تمَّ إنشاءُ جدولين، هما: جدول لتمثيل طلبة درجة البكالوريوس، وجدول لتمثيل طلبة الدراسات العليا. أمَّا فيما يتعلق بالنوع الفرعي «خريج» (ALUMNUS)؛ فإنه لا يرتبط بأيِّ أنواع فرعية ولكنه يرتبط بخاصية مركبة ومتعددة القيم في ذات الوقت، وهي «الدرجة العلمية» (Degree) التي تمَّ تحويلها حسب قاعدة التحويل رقم (1).

5–1–8 قاعدة التحويل الثامنة: التعامل مع التجميع:

عند وجود تجميع؛ يتمُّ تحويل الكينونات والعلاقات المجمَّعة حسب قواعد التحويل التي سبق شرحها أعلاه. أمَّا بالنسبة لعلاقة التجميع التي تربط بين كينونةٍ ما، من جانب، والتجميع، من جانب آخر؛ فيتمُّ التعامل معها وكأنها علاقةٌ تربط بين كينونتين. فعلى سبيل المثال: لنفترض وجود التجميع الممثَّل في الشكل رقم (5-19). إنَّ هذا التجميع (المُمثَّل داخل الشكل المستطيل ذي الخط المتقطع) يربط بين كينونة «قسم» وكينونة «مشروع»؛ من خلال علاقة «دعم مالي». ولتحويل هذا التجميع نستخدم قاعدة التحويل رقم (5-1-3-2) التي توضِّح طريقة تحويل العلاقات الثنائية ذات التعدُّدية «متعدد - متعدد». وباستخدام هذه الطريقة؛ يتمُّ إنشاء ثلاثة جداول، هي: جدولان لتمثيل الكينونتين اللتين تربط بينهما العلاقة الثنائية «دعم مالي» (SPONSOR)، والجدول الثالث لتمثيل العلاقة نفسها. وينتجُ عن هذه الخطوة ثلاثة جداول، هي: جدول القسم (DEPARTMENT)، وجدول المشروع (PROJECT)، وجدول الدعم المالي (SPONSOR). ويوضِّح الشكل المفاتيح الرئيسية والمفاتيح الخارجية لهذه الجداول.



شكل رقم (5–19): تحويل علاقات التجميع إلى النموذج العلاقي.

أمَّا عملية تحويل علاقة التجميع «متابعة» (Monitors)؛ فهي شبيهةٌ بعملية تحويل العلاقات الثنائية التي تربط بين كينونتين؛ إذ يتمُّ إنشاء جدول خاص بالعلاقة يحتوي على حقل لتمثيل المفتاح الرئيسي لجدول كينونة الموظفين وهو «رقم الموظف» (Employee_No)، وحقول المفتاح الرئيسي لجدول علاقة «دعم المادي»، وهي «رقم القسم» (Dept_No) و«رقم المشروع» (ProjeCt_No)، كما يحتوي على حقل لتمثيل الخاصية المرتبطة بالعلاقة، وهي «حتى» (Until). وتُعرَّف الحقول الثلاثة للمفاتيح الرئيسية مجتمعةً بأنها المفتاح الرئيسي لجدول المتابعة، كما يُعَرَّف كلُّ مفتاح على حِدَة بأنه مفتاح خارجي. فالمفتاح (Employee_No) يُعَدُّ جزءاً من المفتاح الرئيسي لجدول العلاقة، وفي الوقت نفسه يُعَدُّ مفتاحاً خارجياً يشير لجدول الموظف. أمَّا حقل «رقم القسم» وحقل «رقم المشروع» فهما مجتمعَيْن يُعدَّان جزءاً من المفتاح الرئيسي للعلاقة، وفي الوقت نفسه يُعدَّان مفتاحاً خارجياً يشير لجدول علاقة الدَّعم المادي.

وهنالك بعض الحالات الخاصة التي تمكِّننا من تحسين عملية التحويل؛ وذلك من خلال حذف الجدول الذي يربط بين الكينونتين المجمعتين. ففي مثالنا السابق؛ يمكن حذف جدول الدعم المالي لو لم ترتبط علاقة «دعم مالي» بخاصية خاصة فيها. غير أنه بشكلٍ عام لا يمكن التحسين على التصميم السابق ما لم يتحقق الشرطان التاليان:

1- ألَّا ترتبط العلاقة التي بين الكينونات المجمَّعة بخصائص خاصة فيها.

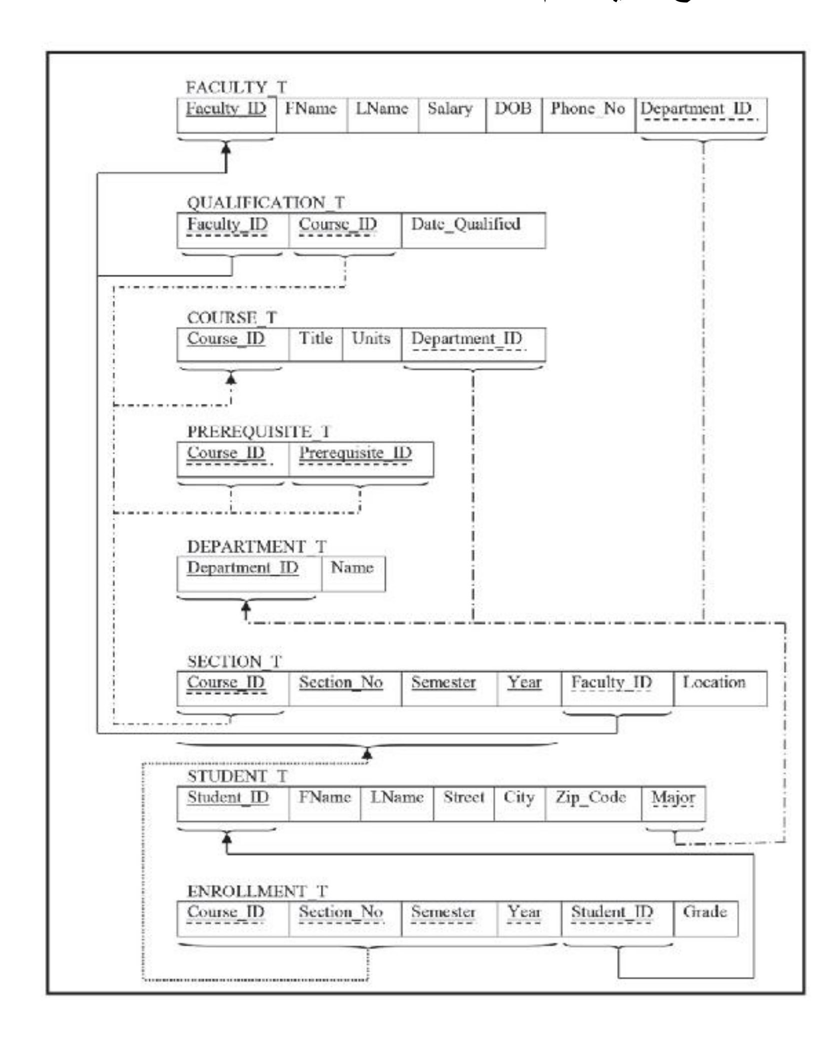
2- أن تكون كلُّ حالة مجمعة مرتبطة بعلاقة التجميع.

ففي مثالنا السابق؛ كل «دعم مالي» يجب أن يرتبط بموظف واحد على الأقل. وبناءً على ذلك؛ فإن هذا الشرط الثاني منطبقٌ على مثالنا، ولكن الشرط الأول غير منطبق. لذا؛ فإننا لا نستطيع إلغاء جدول الدعم المالي من تصميم الجداول العلاقية.

2–5 التصميم المنطقي للحالة الدراسية:

على الرَّغم من أن أدوات هندسة البرمجيات تقوم بتحويل النموذج المفاهيمي إلى النموذج العلاقي بشكلٍ تلقائي؛ فإنه من الأهمية التعرُّف على خطوات التحويل هذه. لذا؛ فإن هذا الفصل قد ركز على قواعد تحويل النموذج المفاهيمي إلى النموذج العلاقي في خطوة تُدعَى «التصميم

المنطقي» لقواعد البيانات. وهذه الخطوة تترجم تصميم قاعدة البيانات من نموذج عالي المستوى، قريب من مستوى إدراك المستفيدين من قاعدة البيانات لبياناتهم، إلى تصميم لقاعدة البيانات نفسها؛ ولكن للنموذج الذي سيتمُّ بناء قاعدة البيانات عليه.



شكل رقم (5–20): التصميم المنطقي الكامل لقاعدة بيانات الجامعة الأهلية.

وبناءً على خطوات التحويل التي تمَّ شرحها في هذا الفصل؛ فإن الشكل رقم (5-20) يوضِّح التصميم المنطقي الكامل لقاعدة بيانات الجامعة الأهلية. ويُلاحَظ في الشكل إضافة الحرف "T" بعد اسم كلِّ جدول؛ وذلك للتفريق بين جداول قاعدة البيانات وبقية أنواع هياكل قاعدة البيانات، مثل: الفهارس والمنظورات – التي سيتمُّ التطرُّق إليها في الفصول المتعلقة بلغة الاستفسار البنائية (الفصلان السابع والثامن). ويُلاحَظ في الشكل أيضاً استخدام خطوط مختلفة بعضها متصل والبعض الآخر منقط؛ غير أن هذا الاختلاف في طبيعة الخطوط لا يدل على اختلافٍ في المعنى

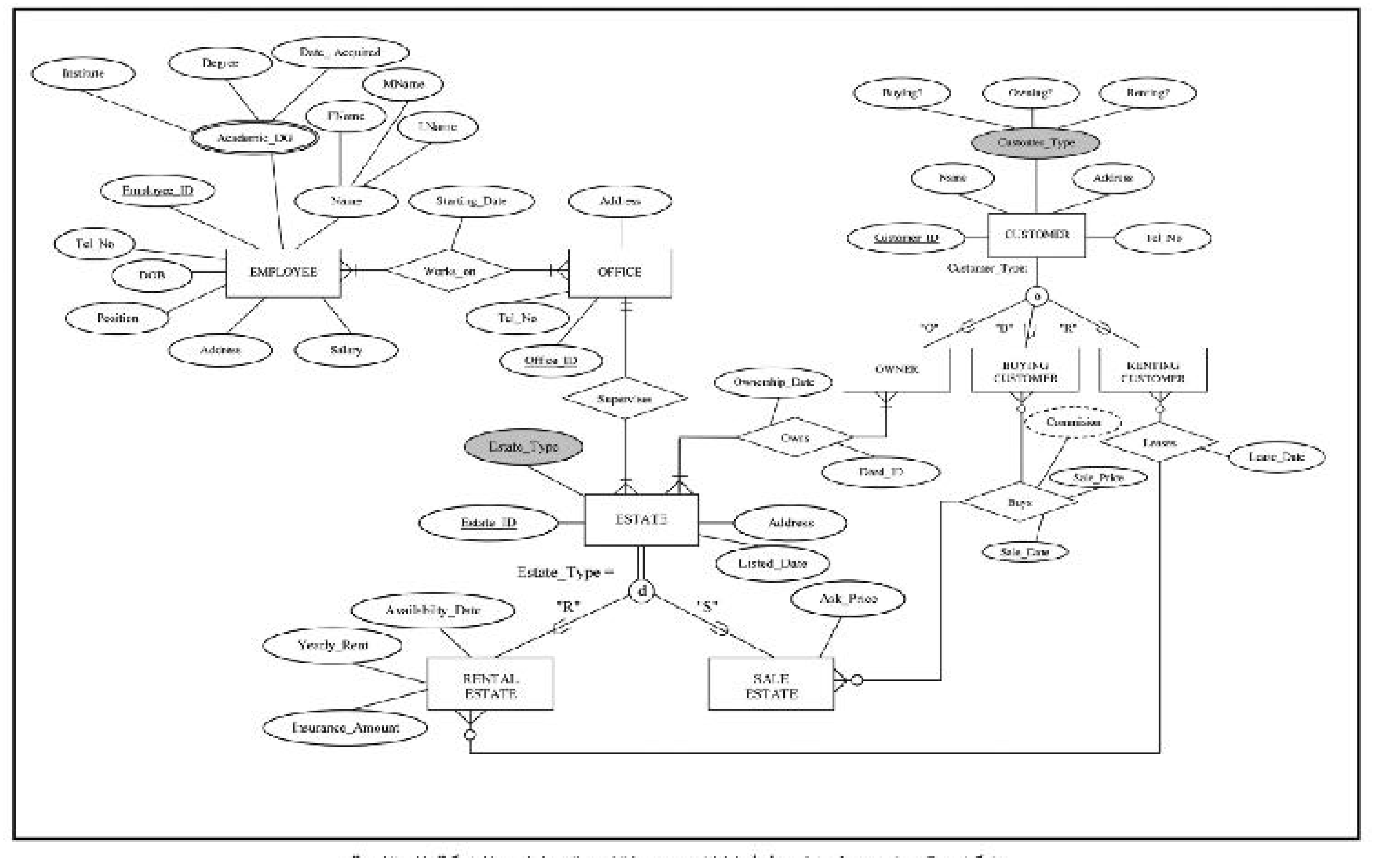
المقصود بها؛ إذ إنها جميعاً تستهدف ربط المفاتيح الخارجية بالمفاتيح الرئيسية التي تشير إليها، وأن هذا الاختلاف في طبيعة الخطوط جاء بشكلٍ مُتعمدٍ حتى تسهل عملية تتبع الخطوط في الشكل فقط.

حالة دراسية

قاعدة بيانات شركة عقارية

(الحل متوفر في الملحق رقم 2)

على افتراض المخطط المفاهيمي في الشكل رقم (5-21) الذي يمثل قواعد العمل المعمول بها في الشركة العقارية التي سبق عرضها في الفصل الثالث؛ المطلوب تحويل المخطط المفاهيمي إلى مخطط منطقي (جداول علاقية)، وإيضاح كافة تفاصيله.



شكل رقم (٥-٢١): المخطط المفاهيمي لقاعدة بيانات الشركة العقارية

الفصل السادس

تطبيع العلاقات والتصميم المادي
لقواعد البيانات العلاقية

سبق أن أشرنا في الفصل السابق أن مرحلة التصميم المنطقي لقواعد البيانات تتكوَّن من خطوتين رئيسيتين: في الخطوة الأولى يتمُّ تحويل النموذج المفاهيمي إلى نموذج قاعدة البيانات المستخدمة، وهو النموذج العلاقي الذي يمثل أحد محاور هذا الكتاب. وقد تم شرح هذه الخطوة في الفصل السابق. أمَّا في الخطوة الثانية؛ فيتمُّ تحسين تصميم قاعدة البيانات الناتجة من عملية التحويل؛ بحيث تحتوي على أقل قدرٍ ممكنٍ من البيانات المتكررة؛ حتى يتمَّ تجنُّب المشكلات التي قد تنتج عن عمليات التعديل على محتويات قاعدة البيانات. وتدعى هذه الخطوة بعملية «التطبيع» (Normalization)، التي تمثل موضوع الجزء الأول من هذا الفصل.

أمَّا الجزء الثاني من هذا الفصل؛ فيركز على مرحلة التصميم المادي لنظم قواعد البيانات الذي يهدف إلى إنشاء تصميم يُمكِّن من تخزين البيانات بشكلٍ يوفر الأداء المناسب لنظام إدارة قاعدة البيانات على اختلاف حجم العمليات التي تُنفَّذ عليها. ويعني هذا؛ وعلى خلاف التصميم المفاهيمي والتصميم المنطقي، أن التصميم المادي يوضِّح الكيفية التي ستُخزَّن وتُعالَج فيها البيانات، لا على الكيفية التي يتمُّ من خلالها التعرُّف على البيانات والعلاقات فيما بينها أو طريقة تمثيلها وفق النموذج العلاقي أو نماذج البيانات الأخرى.

1-6 التطبيع (Normalization):

عند شرح النموذج المفاهيمي، أعملنا الحَدْس في أثناء عملية التعرُّف على الكينونات وتجميع الخصائص التابعة لكلٍّ منها. وبعد ذلك؛ تمَّ استخدام خطوات مُحدَّدة لتحويل النموذج المفاهيمي إلى علاقات. إلا أن الاستناد إلى الحَدْس فقط في تصميم قواعد البيانات غير كافٍ ولا يمكِّننا من قياس أو معرفة جودة الجداول المكوِّنة لقاعدة البيانات. لذلك؛ فإننا بحاجةٍ إلى طريقة رسمية واضحة المعالم والأسس النظرية التي تمكِّننا من معرفة جودة الجداول التي تمَّ تصميمها. وهذه الطريقة الرسمية تُسمَّى «التطبيع»

(Normalization). ولكن قبل البدء في التعرُّف على مفهوم التطبيع والخطوات التي تُتبع للتأكُّد من جودة جداول قاعدة البيانات؛ سنقوم بإيضاح المقصود بالجداول جيدة البناء (Well-StruCtured Relations).

6-1-1 الجداول جيدة البناء (**Well-StruCtured Relations**):

من المنطقي أن يحتوي أيُّ جدولٍ على أقل قدرٍ ممكنٍ من التكرارية؛ وذلك لأن تكرارية البيانات، كما سبق أن أوضحنا في الفصل الأول؛ تؤدي إلى مشكلات أو عدم تناسق في البيانات ما لم يتم التعرُّف على مكامنها بشكلٍ دقيق والتحكُّم فيها بشكلٍ كامل. لذا؛ فإن أيَّ جدول يجب أن يحتوي على أقل قدر ممكن من التكرارية في بياناته؛ بحيث يُمكِّن المستفيدين من التعامل مع محتوياته، من خلال عمليات الحذف والتحديث والإضافة، دون حدوث مشكلات أو عدم تناسق في البيانات. ويمثل الجدول رقم (6-1) جدولاً جيد البناء؛ لأن كلَّ صفٍّ فيه يمثل البيانات المتعلقة بعضو هيئة تدريس واحد ودون وجود أية تكرارية في بيانات عضو هيئة التدريس. كما أن بإمكان المستفيدين من الجدول حذف أيِّ سجل فيه أو تحديث أيِّ حقل من حقوله، أو إضافة أيِّ سجلٍ جديدٍ دون أية مشكلات (أو عدم تناسق) في بياناته؛ وذلك لأن أيًّا من هذه التغييرات محصورةٌ في سجل واحد من سجلات الجدول.

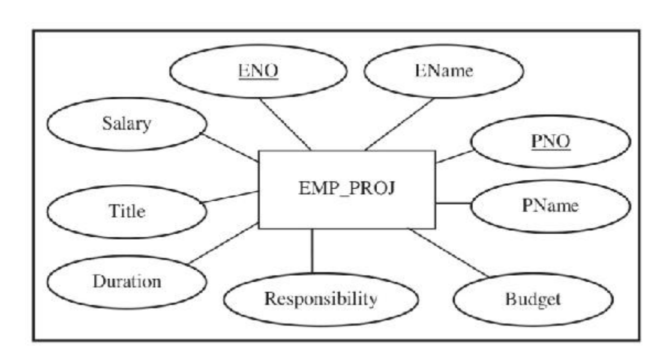
جدول رقم (6-1): مثال لجدول جيد البناء.

FACULTY_T

Faculty_ID	FName	LName	Phone_NO	Salary	DOB
200	Khalid	Aloufi	454-2341	35000	22/05/1963
220	Fahad	Alhamid	456-7733	25900	07/10/1970
310	Saleh	Aleesa	454-8932	30000	13/09/1966
320	Mohammed	Alhamad	454-5412	44000	13/05/1965
330	Ghanim	Alghanim	456-2234	44500	12/08/1969
340	Ibraheem	Alsaleh	454-1234	25000	20/01/1970
400	Ahmad	Alotaibi	454-4563	33900	17/05/1971

على النقيض من الجدول السابق؛ فإن التصميم المُمثَّل بالنموذج المفاهيمي في الشكل رقم (6-1) يُعَدُّ تصميماً سيئاً؛ حيث سينتجُ عنه جدول سيئ البناء أيضاً. والسبب وراء ذلك أن الجدول رقم (6-2) الناتج عن هذا التصميم؛ يحتوي على الكثير من التكرارية في بياناته. فعلى سبيل المثال: تتكرر البيانات الخاصة باسم الموظف، ومُسمَّى الوظيفة، والراتب لكلٍّ من الموظف رقم "E2" والموظف رقم "E3" في صفين من صفوف الجدول. ونتيجةً لذلك؛ فإننا لو حاولنا تعديل راتب أو رقم هاتف أيٍّ من هذين الموظفين؛ فإنه يجب علينا إجراء التحديث في سجلين من سجلات الجدول عوضاً عن سجل واحد (كما هو الحال في الجدول رقم

(1-6)). ونتيجةً لهذه التكرارية في بيانات الجدول؛ فإنه من الممكن أن ينتج عن عمليات التعديل عليه مشكلات (أو عدم تناسق) في البيانات. ويُوجَد هناك ثلاثة أنواع من مشكلات التعديل (ModifiCation Anomalies). وهذه المشكلات تتمثل في مشكلة الإضافة (Insertion Anomaly)، ومشكلة الحذف (Deletion Anomaly)، ومشكلة التحديث (Update Anomaly)، كما يلي:



شكل رقم (6–1): مثال لتصميم مفاهيمي سيئ.

1- مشكلة الإضافة: لو أردنا إضافة سجل لموظف جديد؛ فإننا يجب أن نضيف قيمةً لحقل «رقم المشروع» (PNO) بالإضافة إلى بيانات الحقول المتعلقة بالموظف؛ وذلك لأن حقل رقم المشروع يُعَدُّ جزءاً من المفتاح الرئيسي للجدول، ولا يمكن أن تكون قيمته غائبة. لذلك؛ فإن هذا الجدول يحتوي على مشكلة، وإن هذه المشكلة تتسبب في عدم إمكانية إضافة سجلات جديدة للموظفين إلا بإضافة بيانات تتعلق بالمشاريع.

2- مشكلة الحذف: لو قمنا بحذف سجل الموظف رقم "E1"؛ فإننا لن نحذف البيانات المتعلقة بهذا الموظف فحسب؛ ولكنه سيتم حذف البيانات المتعلقة بالمشروع رقم "P1" كذلك. لذلك؛ فإن هذا الجدول يحتوي على مشكلة، وإن هذه المشكلة تتسبب في عدم إمكانية حذفنا لبيانات الموظفين دون حذف بيانات تتعلق بالمشاريع.

3- مشكلة التحديث: لو أردنا تغيير أرقام هواتف أو رواتب أيٍّ من الموظفين رقم "E2" ورقم "E3"؛ فإنه يجب علينا إجراء مثل عمليات التحديث هذه في أكثر من سجل، وإلا أصبحت حقول بيانات الجدول غير متناسقة في محتوياتها.

جدول رقم (6–2): مثال لجدول سيئ البناء ناتج عن تصميم مفاهيمي سيئ.

EMP_PROJ

ENO	EName	Title	Salary	PNO	PName	Budget	Duration	Responsibility
E1	Saleh Aloufi	Electrical Eng	40000	P1	Database System	170000	12	Manager
E2	Ahmad Alhamid	System Analyst	35000	P3	Human Resources	190000	20	System Analyst
E2	Ahmad Alhamid	System Analyst	35000	P2	Inventory	220000	10	System Analyst
E3	Mohamed Alhamad	Mechanical Eng.	37000	P3	Human Resources	190000	10	Consultant
E3	Mohamed Alhamad	Mechanical Eng.	37000	P4	Maintenance	230000	14	Consultant
E4	Khalid Alsalek	Programmer	29000	P2	Inventory	220000	26	Programmer
E5	Ibraheem Alotaibi	Database Admin.	45000	P2	Inventory	220000	14	Manager
E6	Mishal Aleesa	Programmer	29000	P4	Maintenance	230000	16	Programmer
E7	Abdullah Alghanim	Programmer	29000	P3	Human Resources	190000	12	Manager
E8	Turky Alsalman	Secretary	25000	P3	Human Resources	190000	12	Project Secretary

وتدلُّ المشكلات السابقة في الجدول على أنه سيئ التصميم؛ وذلك لكونه يحتوي على بيانات تتعلق بـشيئين مختلفين؛ أحدهما هو «الموظف»، والثاني هو «المشروع». وللتعرُّف على وجود مثل هذه المشكلات في أيِّ جدول ومعالجتها؛ فإننا نستخدم تطبـيع الجدول.

2-1-6 مستويات التطبيع:

إنَّ عملية تطبيع العلاقات (أو الجداول) «عملية رسمية» (Formal ProCess) تمكِّننا من التعرُّف على مكامن المشكلات في الجداول التي تمَّ تصميمها في أثناء الخطوة الأولى من عملية التصميم المنطقي؛ وذلك قبل الانتقال إلى مرحلة التصميم المادي الذي نقوم من خلالها ببناء قاعدة البـيانات. لذا؛ فإن التطبيع يُعَدُّ أداةً لتحسين تصميم الجداول الناتجة من الخطوة الأولى للتصميم المنطقي؛ بحيث تتحقق عليها بعض الشروط التي تمنع من التكرارية غير المرغوب فيها للبـيانات. وفي أثناء عملية التطبيع؛ يتمُّ اختبار كل جدول للتأكد من تحقيقه لشروط أحد الأشكال الطبـيعية. ويتمُّ في أثناء عملية التطبيع النظر في تصميم كلِّ جدول وتجزئته إلى أكثر من جدول؛ بغية تحسين تصميم الجدول الأساسي ليتوافق مع الخصائص المطلوب أن يتحلى بـها الجدول. لذا؛ فإن عملية التطبيع تُعَدُّ عمليةً هرميةً من الأعلى إلى الأسفل تهدف إلى الفصل بين المفاهيم (أو الأشياء) التي نقوم بـنمذجتها. ويوضِّح الشكل رقم (6-2) مستويات الأشكال الطبيعية؛ بحيث أنه كلما زاد رقم الشكل الطبيعي (وصولاً إلى الداخل)؛ كانت الشروط المصاحبة للشكل الطبيعي أكثر شدةً من الشكل الطبيعي الذي يسبقه، وبحيث يقلل من تكرارية البيانات التي يقبلها الشكل الطبيعي الذي قبله.

All Relations (Tables) كافة العلاقات (الجداول)

1NF الشكل الطبيعي الأول

2NF الشكل الطبيعي الثاني

3NF الشكل الطبيعي الثالث

BCNF الشكل الطبيعي بويس-كود

4NF الشكل الطبيعي الرابع

- - - -

شكل رقم (6–2): مستويات تطبيع العلاقات (أو الجداول).

6–1–3 الاعتماديات الوظيفية (FunCtional DependenCies (FDs)):

تعتمد عملية تطبيع الجداول على ما يُعرَف بالاعتماديات الوظيفية (DependenCies FunCtional). والاعتمادية الوظيفية؛ هي قيدٌ بين حقلين أو مجموعتين من الحقول في الجدول؛ بحيث إن أحد الحقلين أو إحدى المجموعتين تحَدِّد وبشكلٍ منفرد الحقل أو المجموعة الأخرى من الحقول، وفي أية حالة من الحالات التي قد يكون عليها الجدول. ويعني هذا أن الحقل الواحد؛ قد يعتمد وظيفياً على حقلين أو أكثر من حقول الجدول. ففي جدول «الموظف ـ المشروع» (EMP_PROJ) أعلاه؛ يعتمد كلٌّ من حقل «المدة» (Duration)، وحقل «المسئولية» (Responsibility) وظيفياً على حقلي «رقم الموظف» (ENO) وحقل «رقم المشروع» (PNO) مدمجين مع بعضهما. ويعني هذا أن قيمة حقل «رقم الموظف» وقيمة حقل «رقم المشروع» مجتمعين يُحدِّدان قيمة كلٍّ من حقل «المدة» وقيمة حقل «المسئولية» بشكلٍ منفردٍ في جميع سجلات الجدول سواء تلك المدوَّنة فيه فعلياً أو تلك التي قد تدوَّن فيه مستقبلاً. ويتمُّ تمثيل مثل هاتين الاعتماديتين الوظيفيتين، كما يلي:

$$(ENO, PNO) \rightarrow Duration$$

$$(ENO, PNO) \rightarrow Responsibility$$

وتعني الاعتمادية الوظيفية الأولى أنه يمكن معرفة (أو تحديد) الفترة الزمنية التي عمل فيها أيُّ موظف على أيِّ مشروع، وفي أيَّة حالة يكون عليها محتوى الجدول، من خلال معرفة رقم الموظف ورقم المشروع. أمَّا الاعتمادية الثانية؛ فتعني أن مسئولية أيِّ موظف في أيِّ مشروع يمكن معرفتها من خلال رقم

الموظف ورقم المشروع. ويُلاحَظ هنا أنه لا يمكن تحديد «المدة» أو «المسئولية» من خلال معرفة رقم الموظف أو رقم المشروع فحسب، ولكنه يجب معرفة الاثنين معاً لتحديد كلٍّ من «المدة» و«المسئولية» بـشكلٍ منفرد. كما يمكن تمثيل الاعتماديتين الوظيفيتين أعلاه، كما يلي:

(ENO, PNO) → (Duration, Responsibility)

ويعني التمثيل أعلاه؛ أن الحقول الواقعة في الجهة اليسرى من السهم، وتُدعَى المحَدِّدات (Determinants)، تُحدِّد، وبـشكلٍ منفرد، الحقول الواقعة في الجهة اليمنى من السهم. ففي التمثيل السابـق؛ يُحدِّد الحقلان «رقم الموظف» و«رقم المشروع»، معاً، كلاً من حقل «المدة» وحقل «المسئولية». ومن أمثلة الاعتماديات الوظيفية الأخرى في الجدول، ما يلي:

:(ENO, PNO) → (EName, Title, Salary, PName, Budget, Duration, Responsibility)

يُحدِّد رقم الموظف ورقم المشروع مجتمعين بقية حقول الجدول (وذلك لكونهما المفتاح الرئيسي للجدول).

2- ENO → (EName, Title, Salary): يحَـدِّد رقـم الموظـف كلاً من اسـم الموظف، ومُسمَّى وظيفته، وراتبه.

3- PNO → (PName, Budget): يُحدِّد رقم المشروع اسم المشروع، وميزانية المشروع.

4- Title → Salary: يُحدِّد المُسمَّى الوظيفي للموظف الراتب الذي يتقاضاه الموظف.

ونظراً لأن من خصائص المفتاح الرئيسي لأيّ جدول تحديد سجلات الجدول بشكلٍ منفرد؛ فإنَّ حقلي «رقم الموظف» و«رقم المشروع» يُحدِّدان كلَّ حقلٍ من حقول الجدول بـشكل منفرد. فعلى سبيل المثال: إذا عرفنا أن رقم الموظف هو "E2" وأن رقم المشروع هو "P1"؛ فإن هاتين القيمتين تُحدِّدان، وبشكلٍ منفرد، بـقية حقول السجل. وتوضِّح الاعتمادية الوظيفية الأولى أعلاه هذا المفهوم. أمَّا الاعتمادية الوظيفية الثانية والثالثة؛ فتُوضِّحان أنه من خلال معرفة قيمة حقل «رقم الموظف» نستطيع معرفة بقية بيانات الموظف، ومن خلال معرفة قيمة حقل «رقم المشروع» نستطيع معرفة بـيانات المشروع. وتُسمَّى مثل هاتين الاعتماديتين اعتماديات وظيفية جزئية (Partial FunCtional DependenCies)؛ لأن الحقول الواقعة في الجهة اليمنى في كلٍّ من الاعتماديتين الوظيفيتين تعتمدُ على جزءٍ من حقول المفتاح الرئيسي وليس جميع حقوله. أمَّا الاعتمادية الوظيفية الرابعة؛ فتوضِّح أن معرفة قيمة حقل «مُسمَّى الوظيفة» للموظف تمكِّننا من معرفة راتبه.

ويمكن تعريف الاعتمادية الوظيفية بشكلٍ رسمي، كما يلي:

إذا افترضنا وجود جدول اسمه "R" يحتوي على عدد من الحقول "A" بحيث إن $A = \{A_1, A_2, ..., A_n\}$، وكانت كلٌّ من "X" و "Y" تمثّل مجموعةً جزئيةً من حقول الجدول ($X \subseteq A,\ Y \subseteq A$)، وإذا كان لأية زوجين من السجلات في الجدول، وليكونا t_1, t_2، في أية حالة صحيحة من حالات الجدول:

$$[t_1[X] = t_2[X] \Longrightarrow t_1[Y] = t_2[Y]$$

فإنه يوجد اعتمادية وظيفية في الجدول "R" بين الحقل أو مجموعة الحقول الممثلة في "X" و "Y"، كما يلي:

$$X \rightarrow Y$$

وتُعَدُّ الاعتمادية الوظيفية الرابعة أعلاه مثالاً جيداً لفهم تعريف الاعتماديات الوظيفية؛ إذ إن تساوي مُسمَّى الوظيفة لأيِّ اثنين من الموظفين يعني بالضرورة تساوي المرتبات التي يتقاضاها كلا الموظفين. وهذا يعني أن مُسمَّى الوظيفة يدل دائماً على الراتب الذي يتقاضاه الموظف، وكذلك هو الحال بالنسبة للاعتمادية الوظيفية الثانية والثالثة؛ فالاعتمادية الثانية تعني أن أيَّ سجلين يحتويان على «رقم الموظف» نفسه ستكون قيمُ كلٍّ من حقل «اسم الموظف» وحقل «مُسمَّاه الوظيفي» وحقل «راتبه» متساوية فيهما. أمَّا الاعتمادية الوظيفية الثالثة؛ فتعني أن أيَّ سجلين يحتويان على «رقم المشروع» نفسه ستكون قِيم كلٍّ من حقل «اسم المشروع» وحقل «ميزانية المشروع» متساوية فيهما.

وبناءً على تعريف الاعتماديات الوظيفية؛ يمكن تعريف المفتاح الخارق، الذي سبق التطرق له (في الجزء 4-1-1-2 من الفصل الرابع)، لأيِّ جدول، وبشكلٍ رسمي، على أنه مجموعة من الحقول تمكِّن من التعرُّف على سجلات الجدول بشكلٍ منفرد، كما يلي:

إذا افترضنا وجود جدول اسمه "R" يحتوي على عددٍ من الحقول "A" بحيث إن $A = \{A_1, A_2, ..., A_n\}$، وأن مجموعة جزئية من حقوله، ولتكن "K" ($K \subseteq A$)، تمثل مفتاحاً خارقاً للجدول؛ فإنه لأية زوجين من السجلات في الجدول، وليكونا t_1, t_2، في أية حالة صحيحة من حالات الجدول يجب أن يتحقق الشرط التالي:

$$t_1[K] = t_2[K] \Longrightarrow t_1 = t_2$$

ويعني التعريف أعلاه أنه لا يمكن أن يكون في أيِّ جدول سجلان مختلفان لهما المفتاح الخارق نفسه؛ لأن تَساوي قيم المفتاح الخارق في سجلين يعني بالضرورة أنهما عبارة عن سجل واحد. أمَّا المفتاح المرشَّح؛ فيكون في هذه الحالة عبارة عن مفتاح خارق؛ ولكنه لا يحتوي على مفتاح خارق آخر بمعنى أنه لا يمكن أن نقوم بحذف أيِّ حقل من حقوله مع الاستمرار في التعرُّف على سجلات الجدول بشكلٍ منفرد، كما يوضح التعريف الرسمي التالي:

إذا افترضنا وجود جدول اسمه "R" يحتوي على عددٍ من الحقول "A" بحيث إن $A = \{A_1, A_2, \ldots, A_n\}$، وأن مجموعة جزئية من حقوله، ولتكن "C" $(C \subseteq A)$، تمثل مفتاحاً مرشحاً للجدول؛ فإنه لأية زوجين من السجلات في الجدول، وليكونا t_1, t_2، في أية حالة صحيحة من حالات الجدول يجب أن يتحقق الشرط التالي:
$t_1[C\text{-}A_i] = t_2[C\text{-}A_i] \not\Rightarrow t_1 = t_2$

ويعني التعريف أعلاه أنه إذا تساوى سجلان من سجلات أيِّ جدول في قِيَم بعض حقول المفتاح المرشح؛ فإن هذه الحقول لا تعني أن السجلين هما في الواقع يمثلان السجل نفسه، كما هو الحال في تعريف المفتاح الخارق أعلاه.

وتُقسَّم الحقول في أيِّ جدول إلى نوعين: النوع الأول هو الحقول الأولية، والنوع الثاني هو الحقول غير الأولية، كما يلي:

• الحقل الأولي: هو حقل ينتمي لأحد المفاتيح المرشحة.
• الحقل غير الأولي: هو حقل لا ينتمي لأيِّ مفتاح مرشح.

وبناءً على التعاريف والمفاهيم السابقة؛ نقدِّم فيما يلي الأشكال الطبيعة الثلاثة الأولى التي قام «كود» باقتراحها (Codd, 1972) لتصبح سلسلة توصِّلنا إلى الخصائص المرغوب فيها في هياكل الجداول التي يوفرها الشكل الطبيعي الثالث.

1-3-1-6 الشكل الطبيعي الأول (First Normal Form (1NF)):

يُشترَط لأيِّ جدول في شكله الطبيعي الأول أن يحتوي على قيمة واحدة فقط في أيِّ حقل من حقوله؛ مما يعني أنه لا يمكن لأيِّ جدول في شكله الطبيعي الأول أن يحتوي على حقل متعدِّد القيم. ويُعَدُّ الشكل الطبيعي الأول من ضِمن التعريف الرَّسمي لهياكل الجداول العلاقية؛ إذ إنَّ أيَّ جدول علاقي لا يمكن أن

يحتوي على حقول متعددة القيم. وقد تمَّ تعريف هذا الشكل الطبيعي تاريخياً لتأكيد أن الجداول العلاقية يجب ألا تحتوي على حقول متعددة القيم.

ولإيضاح طريقة تطبيع الجداول إلى الشكل الطبيعي الأول لنفترض الجدول رقم (6-3) الذي يتكون من أربعة حقول، هي: حقل «رقم القسم»، وحقل «اسم القسم»، وحقل «رقم الموظف» الذي يرأس القسم، وحقل «الموقع»، ولنفترض أيضاً وجود أكثر من موقع لبعض الأقسام.

جدول رقم (6-3): جدول ليس في الشكل الطبيعي الأول.

DEPARTMENT_T

DNO	DName	D_MGR_NO	Location
10	Research	10101010	(Riyadh)
20	Computer Center	20202020	(Riyadh, Jeddah, Dammam)
30	Administration	30303030	(Riyadh)

إن الجدول رقم (6-3) ليس بالشكل الطبيعي الأول؛ لأن حقل الموقع في السجل الثاني يحتوي على أكثر من قيمة. ويمكِن تفسير محتويات حقل «الموقع» وفق أحد التفسيرين التاليين:

1- مجال حقل «الموقع» مكوَّن من قيم غير مركبة (وهي أسماء المدن)؛ ولكن الحقل قد يحتوي على أكثر من قيمة. ويعني هذا أن حقل «الموقع» لا يعتمد وظيفياً على حقل «رقم القسم». والسبب وراء ذلك أن قيمة المفتاح الرئيسي لا يمكن أن تحَدِّد قيمة واحدة لحقل «الموقع».

2- مجال حقل «الموقع» يتكوَّن من مجموعة من القِيم؛ وبذلك فهو ذو قيم مركبة. وفي هذه الحالة؛ فإن حقل «الموقع» يعتمد وظيفياً على المفتاح الرئيسي للجدول.

ووفقاً لكلا التفسيرين السابقين لحقل «الموقع» لا يُعَدُّ الجدول رقم (6-3) بالشكل الطبيعي الأول (أو جدولاً علاقياً). ولتطبيع الجدول حتى يصبح بالشكل الطبيعي الأول، يُوجَد ثلاث طرق، وهي كما يلي:

1- إزالة حقل «الموقع» الذي يخالف الشكل الطبيعي الأول ووضعه في جدول جديد. ويُضاف للجدول الجديد حقل المفتاح الرئيسي للجدول الأصلي؛ بحيث يصبح المفتاح الرئيسي للجدول الجديد مكوناً من حقلين هما حقل «رقم القسم» وحقل «الموقع»، وبحيث يوجد سجل لكل موقع من مواقع أيِّ قسم في الجدول الجديد. ويكون الجدولان الناتجان كما هو مُوضَّح في الشكل رقم (6-3).

DEPARTMENT_T

DNO	DName	D_MGR_NO
10	Research	10101010
20	Computer Center	20202020
30	Administration	30303030

DEP_LOCATION_T

DNO	Location
10	Riyadh
20	Riyadh
20	Jeddah
20	Dammam
30	Riyadh

شكل رقم (6–3): نتيجة التطبيع للشكل الطبيعي الأول وفق الطريقة الأولى.

2- إضافة حقل «الموقع» ضِمن المفتاح الرئيسي للجدول الأصلي؛ بحيث يُوجَد سجل لكل موقع من مواقع القسم كما هو موضَّح في الشكل (6-4).

DEPARTMENT_T

DNO	DName	D_MGR_NO	Location
10	Research	10101010	Riyadh
20	Computer Center	20202020	Riyadh
20	Computer Center	20202020	Jeddah
20	Computer Center	20202020	Dammam
30	Administration	30303030	Riyadh

شكل رقم (6–4): نتيجة التطبيع للشكل الطبيعي الأول وفق الطريقة الثانية.

3- إضافة حقول جديدة للجدول الأصلي تساوي الحدَّ الأعلى لعدد مواقع الأقسام المسموح بـه. فلو افترضنا أنَّ الحدَّ الأعلى لعدد مواقع أيِّ قسم هو ثلاثة، يمكن إعادة تصميم الجدول ليصبح متوافقاً مع الشكل الطبيعي الأول وفق هذه الطريقة كما هو مُوضَّح في الشكل رقم (6-5).

DEPARTMENT_T

DNO	DName	D_MGR_NO	Location1	Location2	Location3
10	Research	10101010	Riyadh		
20	Computer Center	20202020	Riyadh	Jeddah	Dammam
30	Administration	30303030	Riyadh		

شكل رقم (6–5): نتيجة التطبيع للشكل الطبيعي الأول وفق الطريقة الثالثة.

إن الخيار الأفضل من ضِمن الخيارات الثلاثة أعلاه هو الخيار الأول؛ لأنه لا يؤدي إلى تكرارية في البيانات كما هو الحال في الخيار الثاني، كما أنه لا يضيع المساحة التخزينية أو يقيِّد الحد الأعلى؛ من المواقع كما هو الحال في الخيار الثالث. إضافةً إلى ذلك؛ لو تمَّ اختيار الطريقة الثانية؛ فإن الجدول سيتم تقسيمه إلى جدولين ليصبح كما في الطريقة الأولى في أثناء عمليات تطبيع الجدول في مراحل لاحقة. أمَّا الطريقة الثالثة؛ فمن عيوبها أيضاً، مقارنةً بالخيار الأول، هو أنها تعقد إجراء عمليات التعامل مع الجدول. فعلى سبيل المثال: كيف ستتمُّ كتابة تعليمة الاستفسار المكافئة للاستفسار التالي: «ما الأقسام التي يُوجَد لها مواقع في مدينة جدة؟».

ويمنع الشكل الطبيعي الأول من أن تكون قيم الحقول متعددة القيم ومركبة في آنٍ واحد. فعلى سبيل المثال: يُوضِّح الجدول رقم (4-6) بيانات الموظفين وبيانات المشاريع التي يعملون عليها. فكلُّ موظف يعمل على عددٍ من المشاريع، وكل مشروع يعمل عليه موظف له رقم وعدد من الأسابيع التي عملها الموظف على المشروع.

جدول رقم (6–4): جدول يحتوي على حقول متعددة القِيم ومركبة.

EMP_PROJ

ENO	EName	Title	Salary	Project	
				PNO	Weeks
E1	Saleh Aloufi	Electrical Eng	40000	P1	12
E2	Ahmad Alhamid	System Analyst	35000	P3	20
				P2	10
E3	Mohamed Alhamad	Mechanical Eng.	37000	P3	10
				P4	14
E4	Khalid Alsaleh	Programmer	29000	P2	26
E5	Ibraheem Alotaibi	Database Admin.	45000	P2	14
E6	Mishal Aleesa	Programmer	29000	P4	16
E7	Abdullah Alghanim	Programmer	29000	P3	12
E8	Turky Alsalman	Secretary	25000	P3	12

في الجدول السابق يعمل كلٌّ من الموظف رقم "E2" والموظف رقم "E3" على مشروعين. وبيانات كلِّ مشروع متعددة القيم؛ لكون كلٍّ من هذين الموظفين يعمل على أكثر من مشروع، وفي الوقت نفسه، مركبة من حقل «رقم المشروع» وحقل «عدد الأسابيع». ويمثل «رقم الموظف» المفتاح الرئيسي للجدول؛ لكونه يميِّز بين سجلات الموظفين المختلفة، في حين يمثل «رقم المشروع» مفتاحاً جزئياً يميِّز بين المشاريع المختلفة التي يعمل عليها الموظف نفسه. ولأن الجدول السابق ليس في الشكل الطبيعي الأول؛ فإنه يمكن تطبيعه ليصبح في الشكل الطبيعي الأول من خلال تجزئته إلى جدولين: جدول خاص ببيانات الموظفين، وجدول خاص ببيانات المشاريع التي تمثل الحقل المتعدد القيم والمركب. ويصبح المفتاح الرئيسي للجدول الجديد، الذي يمثل بيانات المشاريع التي يعمل عليها الموظفون، مكوناً من المفتاح الرئيسي

للجدول الأصلي؛ بالإضافة إلى حقل «رقم المشروع». وبذلك يمكن الربط بين الجدولين ومعرفة بيانات المشاريع التي يعمل عليها كلٌّ موظف. ويمثل الشكل رقم (6-6) الجدولين الناتجين بعد إجراء عملية التطبيع على الجدول الأصلي.

EMPLOYEE_T

ENO	EName	Title	Salary
E1	Saleh Aloufi	Electrical Eng	40000
E2	Ahmad Alhamid	System Analyst	35000
E3	Mohamed Alhamad	Mechanical Eng.	37000
E4	Khalid Alsaleh	Programmer	29000
E5	Ibraheem Alotaibi	Database Admin.	45000
E6	Mishal Aleesa	Programmer	29000
E7	Abdullah Alghanim	Programmer	29000
E8	Turky Alsalman	Secretary	25000

EMP_PROJECTS_T

ENO	PNO	Weeks
E1	P1	12
E2	P3	20
E2	P2	10
E3	P3	10
E3	P4	14
E4	P2	26
E5	P2	14
E6	P4	16
E7	P3	12
E8	P3	12

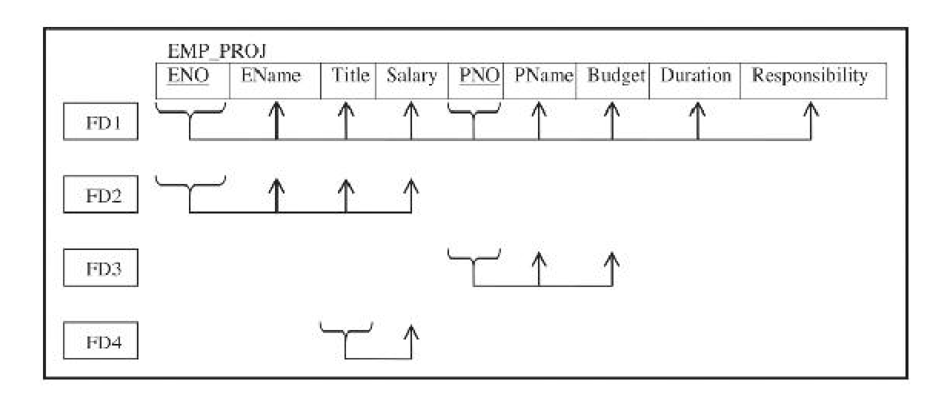
شكل رقم (6–6): تطبيع جدول ذي حقل متعدد القيم ومركب للشكل الطبيعي الأول.

6–1–3–2 الشكل الطبيعي الثاني (SeCond Normal Form (2NF)):

يعتمد الشكل الطبيعي الثاني على مبدأ الاعتمادية الوظيفية الكاملة (Full FunCtional DependenCy). وتُسمَّى أية اعتمادية وظيفية ($X \rightarrow Y$) اعتمادية وظيفية كاملة إذا كان من غير الممكن إزالة أيِّ حقلٍ من الحقول المكوّنة للجانب الأيسر من الاعتمادية مع استمرار تحديد الاعتمادية للجانب الأيمن. ويعني هذا أن عدد حقول الجانب الأيسر يُعَدُّ أقل عدد ممكن من الحقول التي تمكِّن من تحديد الجانب الأيمن في الاعتمادية. أمَّا إذا كان الأمر غير ذلك؛ فإن الاعتمادية الوظيفية تُعَدُّ جزئيةً (Partial FunCtional DependenCy) بمعنى أنه يمكن الاستغناء عن حقل أو أكثر من حقول الجانب الأيسر مع الاستمرار في تحديد الجانب الأيمن في الاعتمادية الوظيفية. ويمكن تعريف هذين النوعين من الاعتماديات الوظيفية بشكلٍ رسمي، كما يلي:

- إذا وجـدت اعتمادية وظيفيـة $X \rightarrow Y$؛ فإنها تُعَدُّ اعتماديـة وظيفية كاملة ($X \xrightarrow{f} Y$) إذا كان مـن غـير الممكن إزالة أيِّ حقـل من الحقـول المكوِّنة للجانب الأيسر مـن الاعتمادية، وهـو A وبحيـث أن ($A \in X$)، مع الاسـتمرار في تحديد الجانب الأيمـن من الاعتمادية ($(X\text{-}\{A\}) \rightarrow Y$).
- إذا وجـدت اعتمادية وظيفيـة $X \rightarrow Y$؛ فإنها تُعَدُّ اعتمادية وظيفية جزئية ($X \xrightarrow{p} Y$) إذا كان مـن الممكن إزالة أيِّ حقلٍ من الحقـول المكوِّنة للجانب الأيسر من الاعتمادية، وهو A وبحيـث أن ($A \in X$)، مع الاسـتمرار في تحديد الجانب الأيمـن من الاعتمادية ($(X\text{-}\{A\}) \rightarrow Y$).

ففي جدول «الموظفين - المشاريع» (EMP_PROJ) الممثل في الجدول رقم (2-6) توجد، كما أسلفنا، الاعتماديات الوظيفية الأربع الممثلة في الشكل التالي:



إن الاعتمادية الوظيفية الأولى أعلاه (FD1) تدلُّ على أن المفتاح الرئيسي المكوَّن من حقل «رقم الموظف» وحقل «رقم المشروع» يحدِّدان قيم الحقول كافة في أيِّ سجل من سجلات الجدول. وهذه الخاصية هي الخاصية الرئيسية للمفتاح الرئيسي؛ إذ إن قيمته تحدِّد السجل المطلوب بشكلٍ منفردٍ وقيم حقوله كافة. أمَّا الاعتمادية الوظيفية الثانية والاعتمادية الوظيفية الثالثة؛ فهي اعتماديات وظيفية جزئية؛ وذلك لأن كلاً منها يمكن اشتقاقها من الاعتمادية الوظيفية الأولى. ففي الاعتمادية الوظيفية الثانية؛ تعتمد قيم حقل «اسم الموظف»، وحقل «مُسمَّاه الوظيفي» (Title)، وحقل «راتبه» على «رقم الموظف» دون الحاجة إلى معرفة «رقم المشروع». ويعني هذا أنه يمكن معرفة قيم هذه الحقول الثلاثة دون معرفة رقم المشروع الذي يعمل عليه الموظف. أما في الاعتمادية الوظيفية الثالثة؛ فتعتمد قيمة حقل «اسم المشروع»، وحقل «ميزانية المشروع» على قيمة «رقم المشروع» دون الحاجة إلى معرفة «رقم الموظف» الذي يعمل فيه. لذا؛ فإن كلاً من الاعتمادية الوظيفية الثانية والاعتمادية الوظيفية الثالثة تُعدَّان اعتماديات وظيفية جزئية من الاعتمادية الوظيفية الأولى. وحسب تعريف الاعتماديات الوظيفية أعلاه؛ فإن الاعتمادية الوظيفية

الأولى تُعَدُّ اعتمادية وظيفية جزئية لوجود اعتماديات وظيفية أخرى يمكن أن تشتق منها. أمَّا الاعتمادية الوظيفية الرابعة؛ فهي اعتمادية وظيفية كاملة؛ إذ لا يمكن أن تشتق أو تستنتج من الوظائف الاعتمادية الأخرى. وبالنظر في البيانات المدوّنة في الجدول نلاحظ أن هذه الاعتماديات، الثانية والثالثة والرابعة، دائماً متحققة.

وقد يطرح السؤال التالي: كيف نستطيع أن نتعرف على الاعتماديات الوظيفية؟ والإجابة هي أن الاعتماديات الوظيفية لا تستنتج من قِبل مُصمِّمي قواعد البيانات من خلال النظر إلى البيانات التي سيتمُّ تخزينها في جداول قاعدة البيانات، وإنما يتم التعرُّف عليها من خلال قواعد العمل في المنظمة، ومن المستفيدين من قاعدة البيانات. فالمستفيدون العاملون في الشئون الإدارية في المنظمة، على سبيل المثال، قد يوضحون أنه من الممكن التعرف على بيانات أيِّ موظف من خلال معرفة رقمه الوظيفي. ويعني هذا وجود اعتمادية وظيفية بين رقم الموظف وبقية البيانات الوظيفية الخاصة فيه. كذلك هو الحال بالنسبة للمرضى المنومين في مستشفى ما؛ فقد يفيد العاملون في المستشفى أن رقم تحويلة هاتف المريض تدل على رقم الغرفة أو السرير المنوم فيه المريض. ويعني هذا أن رقم غرفة أو سرير المريض يعتمد وظيفياً على رقم تحويلة هاتفه.

وللحَدِّ من تكرارية البيانات التي تؤدي إلى مشكلات التعديل؛ يعتمد تعريف الشكل الطبيعي الثاني على عدم وجود أيِّ اعتماديات وظيفية جزئية، كما يلي:

يُعَدُّ هيكل أيَّة جدول علاقي في شكله الطبيعي الثاني (2NF) إذا كان في الشكل الطبيعي الأول، وكان كلُّ حقلٍ من حقوله غير الأولية يعتمد كلياً على المفتاح الرئيسي للجدول.

يعتمد التعريف أعلاه على وجود مفتاح مرشح واحد هو المفتاح الرئيسي، وأن حقوله الأوَّلية هي مجموعة حقول المفتاح الرئيسي فحسب. أمَّا التعريف الأعمُّ للشكل الطبيعي الثاني؛ فيأخذ بعين الاعتبار وجود مفاتيح مرشحة أخرى، ومن ثم وجود حقول أولية غير تلك الحقول التي يتكون منها المفتاح الرئيسي. وهذا التعريف العام، كما يلي:

يُعَدُّ هيكل أية جدول علاقي في شكله الطبيعي الثاني (2NF) إذا كان في الشكل الطبيعي الأول، وكان كلُّ حقل من حقوله غير الأولية (بمعنى أن الحقل ليس جزءًا من أية مفتاح مرشح) يعتمد كلياً على كل مفتاح مرشح للجدول.

ويعني التعريف الأعم للشكل الطبيعي الثاني أن جميع الحقول غير الأولية؛ يجب أن تعتمد وظيفياً على جميع المفاتيح المرشحة، وليس على المفتاح الرئيسي للجدول فقط. وللتحقق من كون أيِّ جدول في شكله الطبيعي الثاني، حسب التعريف الأول، يتم اختبار الاعتماديات الوظيفية المفروضة عليه والتي يكون جانبها الأيسر جزءاً من المفتاح الرئيسي للجدول. ويكون الجدول في شكله الطبيعي الثاني إذا كان في شكله الطبيعي الأول وتحققت فيه أيٌّ من الشروط الثلاثة التالية:

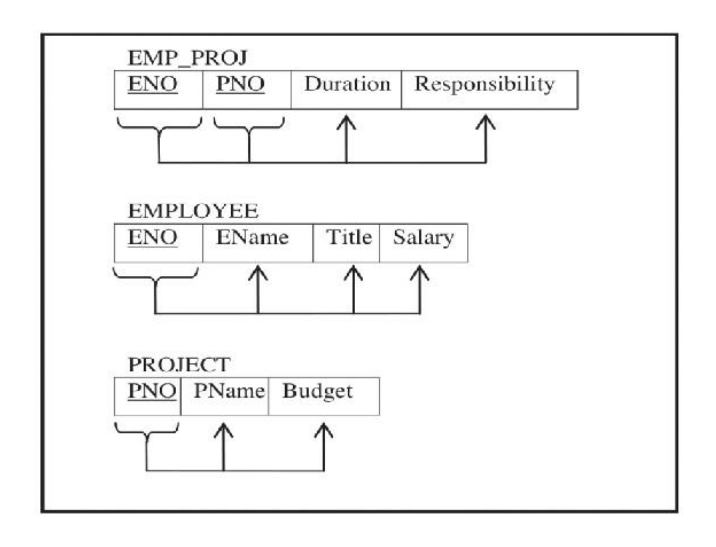
1- المفتاح الرئيسي مكوَّن من حقل واحد فقط.

2- جميع حقول الجدول تُعَدُّ جزءاً من مفتاحه الرئيسي بمعنى عدم وجود أيِّ حقل غير أوَّلي.

3- كلُّ حقل ليس من حقول المفتاح الرئيسي يعتمد على جميع حقول المفتاح الرئيسي، وليس على جزء منها.

أمَّا إذا احتوى الجدول على مفاتيح مرشحة أخرى غير المفتاح الرئيسي له؛ فإنه يجب التحقق من أن كلَّ حقلٍ غير أولي يعتمد كلياً على كل مفتاح مرشح؛ إضافةً إلى اعتماده الكلي على المفتاح الرئيسي.

ويُعَدُّ الجدول «الموظف - المشروع» (EMP_PROJ) المُمثل في جدول رقم (6-2) ليس في الشكل الطبيعي الثاني؛ وذلك بسبب الاعتمادية الوظيفية الثانية والاعتمادية الوظيفية الثالثة اللتين تعتمد أجزاؤهما اليمنى على جزءٍ من حقول المفتاح الرئيسي، وليس حقوله كافة. ولهذا السبب تتكرَّر بعض بيانات الجدول؛ مما يؤدي إلى أخطاء التعديل على الجدول التي سبق أن أوضحناها أعلاه. ولتطبيع الجدول؛ بحيث يصبح في شكله الطبيعي الثاني، تتمُّ تجزئة الجدول إلى اثنين أو أكثر من الجداول؛ بحيث ينطبق على كلٍّ منها أحد الشروط الثلاثة أعلاه. وبمعنى آخر؛ تتمُّ تجزئة الجدول إلى مجموعة من الجداول تنطبق عليها شروط الشكل الطبيعي الثاني. ويتمُّ ذلك من خلال إنشاء جدول جديد لكلِّ اعتمادية وظيفية جزئية حيث يتم إنشاء جدول اسمه «موظف» (EMPLOYEE)، في مثالنا؛ لتمثيل جميع حقول الاعتمادية الوظيفية الثانية، وجدول «مشروع» (PROJECT) لتمثيل جميع حقول الاعتمادية الوظيفية الثالثة، مع الإبقاء على بقية الحقول في الجدول الأساسي دون تغيير لها. وتكون أشكال الجداول الناتجة بعد عملية التجزئة جداول بالشكل الطبيعي الثاني تعتمد حقولها غير الأولية على جميع حقول مفاتيحها الرئيسية، كما يلي:



3-3-1-6 الشكل الطبيعي الثالث (**Third Normal Form** (**3NF**)):

يعتمد تطبيع الجداول إلى الشكل الطبيعي الثالث على مبدأ الاعتمادية الوظيفية الانتقالية (Transitive DependenCy). وتُعَدُّ الاعتمادية الوظيفية ($X \rightarrow Y$) في جدول ما اعتمادية وظيفية انتقالية إذا وجد مجموعة من الحقول، ولتكن (Z) في الجدول، ولا تمثل مفتاحاً مرشحاً للجدول كما أنها ليست مجموعة جزئية من أيِّ مفتاح (سواء أكان مرشحاً أو رئيسياً) للجدول مع وجود الاعتمادية الوظيفية ($X \rightarrow Z$) والاعتمادية الوظيفية ($Z \rightarrow Y$). ويمكن تعريف الاعتمادية الوظيفية الانتقالية بشكلٍ رسمي، كما يلي:

> تُوجَد اعتمادية وظيفية انتقالية $X \rightarrow Y$ إذا تحققت الشروط التالية:
>
> ١. $X \rightarrow Z$
>
> ٢. $Z \rightarrow Y$
>
> ٣. $Z \nrightarrow X$
>
> ٤. $Y \not\subseteq Z$

وتعني الاعتمادية الانتقالية ($X \rightarrow Y$)، بشكلٍ عام؛ أنه يمكن تحديد حقول الجانب الأيمن من الاعتمادية من خلال اعتماديات وظيفية أخرى عوضاً عن هذه الاعتمادية التي تُحدِّد حقول الجانب الأيمن بشكلٍ مباشر.

ولتعريف الشكل الطبيعي الثالث والشكل الطبيعي «بويس-كود»(Boyce-Codd Normal Form (BCNF)) نحتاج إلى تعريف الاعتمادية الوظيفية البديهية، وهو كما يلي:

تُعَــدُّ الاعتمادية الوظيفية $X \rightarrow Y$ بديهيةً إذا كانت الحقول المكوّنة للجانب الأيمن من الاعتمادية الوظيفية مجموعة جزئية أو مساوية لحقول الجانب الأيسر من الاعتمادية ($Y \subseteq X$).

ومثالٌ على الاعتماديات الوظيفية البديهية من جدول «الموظف - المشروع»، تُعَدُّ الاعتماديات الوظيفية التالية بديهية:

1- ENO, PNO → ENO, PNO

2- ENO → ENO, PNO

3- PNO → ENO, PNO

4- PNO → PNO

5- ENO → ENO

وحسب التعريف أعلاه؛ تُعَدُّ كلُّ اعتمادية وظيفية من الاعتماديات الوظيفية الخمس السابقة بديهية؛ لأن حقول الجانب الأيمن هي مجموعة جزئية أو مجموعة مساوية لحقول الجانب الأيسر. على سبيل المثال: تنصُّ الاعتمادية الوظيفية الرابعة على أنَّ رقم المشروع يحدد رقم المشروع، وهو أمرٌ بديهي. كذلك هو الحال بالنسبة للاعتمادية الوظيفية الثانية، على سبيل المثال، التي تنصُّ على أن رقم الموظف ورقم المشروع يحَدِّدان رقم الموظف، وهو أمرٌ بديهي؛ إذ يمكِن تحديد رقم الموظف حتى بدون معرفة رقم المشروع.

وفيما يلي تعريف الشكل الطبيعي الثالث بشكله الأصلي حسب ما اقترحه «كود» (Codd, 1972):

يُعَدُّ هيكل أية جدول علاقي في شكله الطبيعي الثالث (3NF) إذا كان في الشكل الطبيعي الثاني، وكان كلُّ حقل من حقوله غير الأوَّلية لا يعتمد بشكلٍ انتقالي على المفتاح الرئيسي للجدول.

وكما هو الحال بالنسبة للتعريف العام للشكل الطبيعي الثاني، يَفترض التعريف العام للشكل الطبيعي الثالث وجود أكثر من مفتاح مرشح في الجدول، عوضاً عن افتراض وجود مفتاح مرشح واحد وهو المفتاح الرئيسي للجدول. وهذا التعريف العام كما يلي:

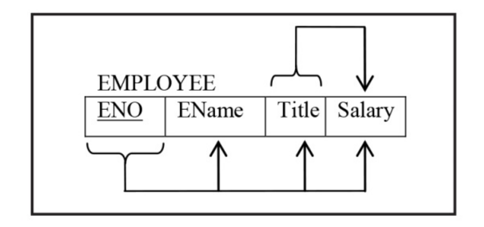
إذا افترضنا وجود هيكل جدول اسمه "R" يحتوي على عددٍ من الحقول "A" بحيث أن $A = \{A_1, A_2, \ldots, A_n\}$، يكـون الجدول في شـكله الطبيعي الثالث (3NF) إذا كانت كافـة الاعتماديـات الوظيفيـة المفروضة عليه بالشـكل $X \rightarrow Y$ وبحيـث أن كلاً من "X" و"Y" تمثل مجموعةً جزئيةً من حقول الجدول $(X \subseteq A, Y \subseteq A)$؛ فإنه يجب أن تتحقق على كل اعتمادية وظيفية، على الأقل، إحدى الشروط التالية:

١. $X \rightarrow Y$ اعتمادية وظيفية بديهية.

٢. X عبارة عن مفتاح خارق للجدول.

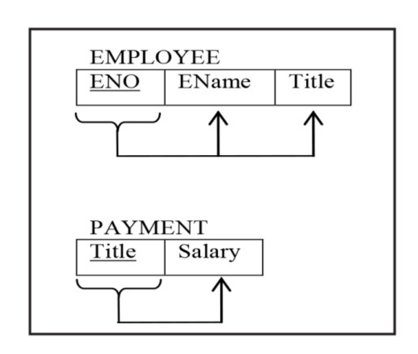
٣. Y عبارة عن حقل أوَّلي (أو مجموعة حقول أولية).

ويُلاحَظ في التعريف العام للشكل الطبيعي الثالث أنه لا ينصُّ على أن يكون الجدول في شكله الطبيعي الثاني؛ وذلك لكون هذا التعريف يمنع وجود اعتماديات وظيفية جزئية، التي ينصُّ على عدم وجودها تعريف الشكل الطبيعي الثاني، بشكلٍ مباشر. عند تطبيق التعريف الأول للشكل الطبيعي الثالث، الذي ينصُّ على أن يكون الجدول في الشكل الطبيعي الثاني، على الجداول الثلاثة التي نتجت بعد عملية تطبيع جدول «مشروع - موظف» إلى الشكل الطبيعي الثاني؛ نجد أنه لا يُوجَد اعتماديات وظيفية انتقالية في كلٍّ من الجدول الأول (EMP_PROJ) والجدول الثالث (PROJECT). لذا؛ فإن هذين الجدولين هما في الشكل الطبيعي الثالث أيضاً. أمَّا الجدول الثاني (EMPLOYEE) فليس في الشكل الطبيعي الثالث؛ وذلك لوجود الاعتمادية الوظيفية الرابعة التي تمثل اعتمادية وظيفية انتقالية؛ إذ إنها تمكِّن من تحديد قيمة حقل «الراتب» ليس من خلال معرفة قيمة حقل المفتاح الرئيسي للجدول، وهو «رقم الموظف»، بشكلٍ مباشر فحسب، ولكن يمكن أيضاً تحديده، بشكل انتقالي، من خلال معرفة «مسمى الوظيفة» (رقم الوظيفة ← مُسمَّى الوظيفة ← الراتب)، كما يلي:



ومثل هذه الاعتمادية الانتقالية تؤدي إلى مشكلات التعديل التي سبق أن أوضحناها بسبب تكرارية البيانات؛ إذ إننا سنجد أنه كلما تكرَّر مُسمَّى وظيفة معينة تكرَّر راتب هذه الوظيفة. وللتغلب على هذه التكرارية؛ يتمُّ تقسيم الجدول إلى أكثر من جدول حسب عدد الاعتماديات الوظيفية الانتقالية؛ بحيث يتمُّ إنشاء جدول جديد لكلِّ اعتمادية وظيفية انتقالية. ولأنه يُوجَد في مثالنا اعتمادية وظيفية انتقالية واحدة فقط؛

فإنه يتم إنشاء جدول جديد اسمه «الأجر» (PAYMENT) تكون حقوله مكوَّنة من الحقول الواردة في الاعتمادية، وهي حقل «مُسمَّى الوظيفة» وحقل «الراتب»؛ بحيث يكون المفتاح الرئيسي للجدول الجديد هو الحقل الموجود (أو مجموعة الحقول الموجودة) في الجانب الأيسر من الاعتمادية، كما يلي:



أمَّا إذا طبقنا التعريف العام للشكل الطبيعي الثالث مباشرةً على أيِّ جدول دون تطبيع الجدول إلى الشكل الطبيعي الثاني؛ فإن هذا التعريف سيُمكِّننا من معرفة الاعتماديات الوظيفية التي تخالف شروط الشكل الطبيعي الثاني؛ إضافةً إلى شروط الشكل الطبيعي الثالث. فعلى سبيل المثال: لو تم تطبيق التعريف العام على جدول «الموظف - المشروع» سنجد أن الاعتمادية الوظيفية الثانية ((ENO → (EName, Title, Salary)) والاعتمادية الوظيفية الثالثة ((PNO → (PName, Budget))تخالفان شروط الشكل الطبيعي الثاني؛ وذلك لكونهما جزئية؛ فهما تخالفان أيضاً شروط الشكل الطبيعي الثالث؛ وذلك لكونهما ليستا بديهيتين، وليست الأجزاء اليسرى منهما تمثل مفاتيح خارقة للجدول، وليست الأجزاء اليمنى منهما تمثل حقولاً أولية. لذا سيتمُّ بناء جداول جديدة لهاتين الاعتماديتين كما هو الحال بالنسبة للاعتمادية الرابعة. وتكون نتيجة عملية التطبيع هذه مماثلةً لعملية تطبيع الجدول إلى الشكل الطبيعي الثاني، ومن ثم تطبيعه إلى الشكل الطبيعي الثالث. ويدل هذا على أنه ليس من الضروري أن تمرَّ عملية التطبيع بالشكل الطبيعي الثاني؛ ولكنه يمكن تطبيع الجدول مباشرة (من الشكل الطبيعي الأول) إلى الشكل الطبيعي الثالث؛ غير أن عملية التسلسل في مراحل التطبيع ذات بعْد تاريخي فقط لكون «كود» قد طرحها بهذه الطريقة.

4-3-1-6 الشكل الطبيعي «بويس - كود» (BoyCe-Codd Normal Form (BCNF)):

تمَّ اقتراح الشكل الطبيعي «بويس - كود» بشكلٍ أساسي على أنه شكلٌ مُبسَّط للشكل الطبيعي الثالث. غير أنه تبيَّن لاحقاً أن هذا الشكل أشد في شروطه من الشكل الطبيعي الثالث؛ بمعنى أنه إذا كان أيُّ جدول في الشكل الطبيعي «بويس - كود»؛ فإنه أيضاً في الشكل الطبيعي الثالث والعكس ليس بالضرورة صحيحاً؛ إذ إن جدولاً ما قد يكون في الشكل الطبيعي الثالث، ولكنه ليس في الشكل الطبيعي «بويس - كود». إن

تعريف الشكل الطبيعي الثالث لا يلغي جميع الاعتماديات الوظيفية الانتقالية وبوجهٍ خاص تلك التي يكون جانبها الأيمن حقلاً أولياً (أو مجموعة حقول أولية). فعلى سبيل المثال: لنفترض أن جدول «الموظف - المشروع» كان يحتوي على حقل يُسمَّى «موقع المشروع» (PLoCation)، وأنَّ كلَّ مشروع قد يكون له عددٌ من المواقع وليس موقعاً واحداً فحسب. ولنفترض أيضاً وجود الاعتماديتين الوظيفيتين التاليتين:

- قد يعمل الموظف في أكثر من مشروع؛ ولكنه عندما يعمل في مشروع، يكون عمله في موقع واحد من مواقع المشروع وأن للموظف مسئوليةً مُحدَّدةً في المشروع وفترة زمنية تُحدِّد فترة عمله في المشروع (ENO, PNO $\rightarrow$ PLocation, Duration, Responsibility).

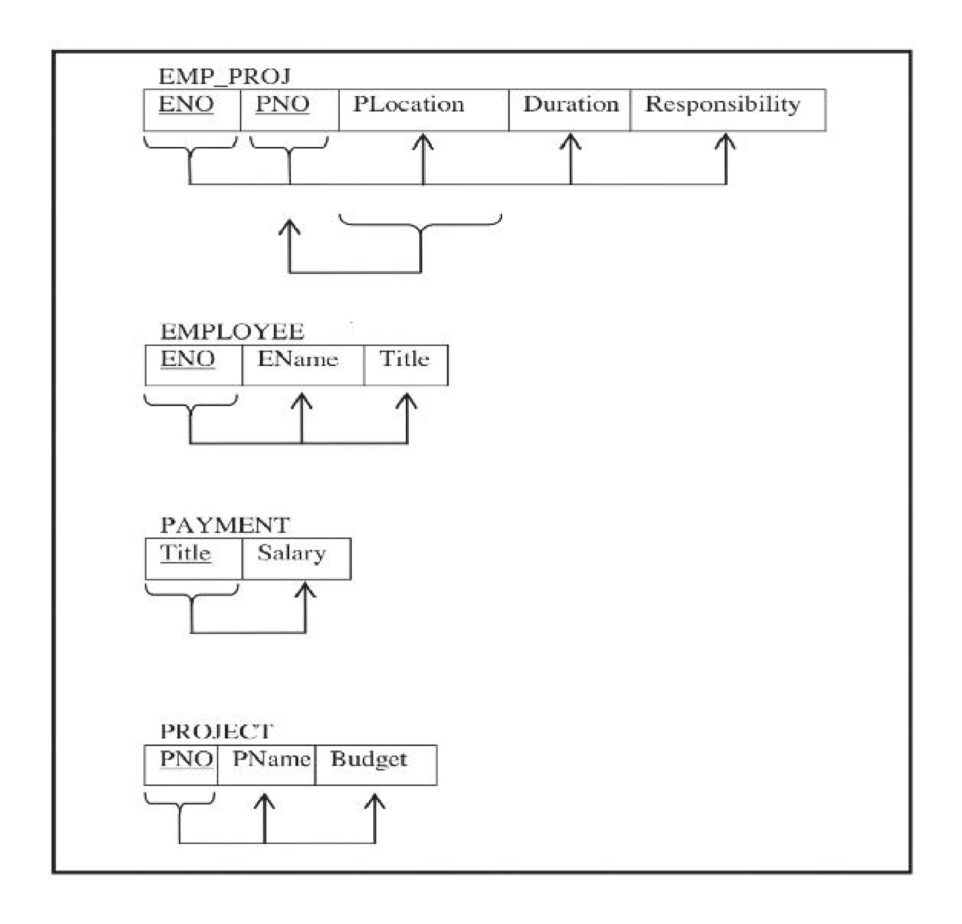
- يُوجَد في أية موقع مشروع واحد فقط (PLocation $\rightarrow$ PNO).

وبعد إضافة الحقل الجديد للجدول وفق الاعتماديتين الوظيفيتين أعلاه، يصبح شكل الجدول، كما يلي:

EMP_PROJ

ENO	EName	Title	Salary	PNO	PName	PLocation	Budget	Duration	Responsibility
E1	Saleh Aloufi	Electrical Eng	40000	P1	Database System	Riyadh	170000	12	Manager
E2	Ahmad Alhamid	System Analyst	35000	P3	Human Resources	Jizan	190000	20	System Analyst
E2	Ahmad Alhamid	System Analyst	35000	P2	Inventory	Jouf	220000	10	System Analyst
E3	Mohamed Alhamad	Mechanical Eng.	37000	P3	Human Resources	Joubail	190000	10	Consultant
E3	Mohamed Alhamad	Mechanical Eng.	37000	P4	Maintenance	Skaka	230000	14	Consultant
E4	Khalid Alsaleh	Programmer	29000	P2	Inventory	Jouf	220000	26	Programmer
E5	Ibraheem Alotaibi	Database Admin.	45000	P2	Inventory	Jouf	220000	14	Manager
E6	Mishal Aleesa	Programmer	29000	P4	Maintenance	Hail	230000	16	Programmer
E7	Abdullah Algharim	Programmer	29000	P3	Human Resources	Jizan	190000	12	Manager
E8	Turky Alsalman	Secretary	25000	P3	Human Resources	Jizan	190000	12	Project Secretary

ونتيجةً للاعتمادية الوظيفية الثانية أعلاه، وكما يُلاحَظ في البيانات المدوَّنة في الجدول؛ يمكِن تحديد رقم أيِّ مشروع من خلال معرفة «الموقع»؛ لأنه لا يمكن أن يُوجَد مشروعان في موقع واحد. وبناءً على تعريف الشكل الطبيعي الثالث؛ تتم تجزئة الجدول إلى أربعة جداول، هي: جدول «الموظف»، وجدول «المشروع»، وجدول «الأجر»، وجدول «الموظف - المشروع» الذي يحتوي على بقية الحقول التي لم تنقل إلى أيٍّ من الجداول الثلاثة الأخرى بالإضافة للمفتاح الرئيسي للجدول المكوَّن من حقل «رقم الموظف» وحقل «رقم المشروع»، كما سبق أن أوضحنا في الجزء السابق؛ لتصبح، كما يلي:



ويُلاحَظ أن الاعتمادية الوظيفية التي تحَدِّد رقم المشروع من خلال الموقع، في الجدول الأول أعلاه؛ تُعَدُّ انتقاليةً ولكنها لا تخالف شروط الشكل الطبيعي الثالث؛ لكون الجانب الأيمن منها يمثل حقلاً أولياً (جزءاً من المفتاح الرئيسي للجدول في هذه الحالة). لذا؛ فإن الجدول «الموظف - المشروع» يُعَدُّ في الشكل الطبيعي الثالث. إلا أن وجود هذه الاعتمادية فيه يؤدي إلى تكرارية في بياناته. فعلى سبيل المثال: لو افترضنا وجود عشرين موقعاً لخمسة من المشاريع التي تقوم المنظمة بالتعامل معها (أو متابعتها) ووجود عشرة آلاف موظف يعملون في هذه المشاريع الخمسة؛ فإن هذين الحقلين ستكرر قيمهما بشكلٍ كبير ضمن الجدول. وللحَدِّ من هذه التكرارية في بيانات الجدول؛ يمكن إنشاء جدول جديد يحتوي على قيم أرقام المشاريع الخمسة، والعشرين موقعاً التي توجد فيها هذه المشاريع. لذا؛ فإن التعريف التالي للشكل الطبيعي «بويس - كود» يحدُّ من مثل تكرارية البيانات هذه ويُعَدُّ أشد في شروطه من شروط الشكل الطبيعي الثالث:

إذا افترضنا وجود هيكل جدول اسمه "R" يحتوي على عدد من الحقول "A"؛ بحيث إن $A = (A_1, A_2, \ldots, A_n)$، يكون الجدول في شــكله الطبيعي «بويس - كود» (BCNF) إذا كانت كافة الاعتماديات الوظيفية المفروضة عليه بالشكل $X \rightarrow Y$، وبحيث إن كلاً من "X" و"Y" تمثل مجموعةً جزئيةً من حقول الجدول ($X \subseteq A, Y \subseteq A$)؛ فإنه يجب أن تتحقق على كل اعتمادية وظيفية، على الأقل، إحدى الشروط التالية:

١. $X \rightarrow Y$ اعتمادية وظيفية بديهية.

٢. X عبارة عن مفتاح خارق للجدول.

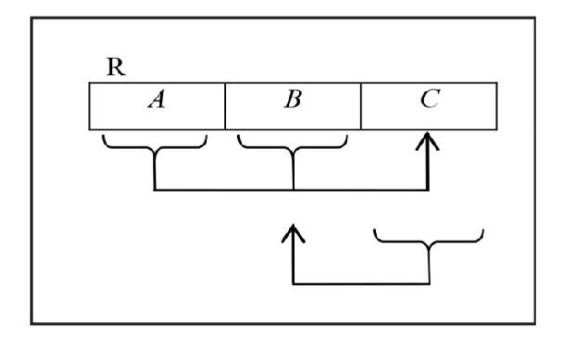
تنطبق شروط الشكل الطبيعي «بويس - كود» على الجدول الثاني والثالث والرابع؛ لأن أيَّ اعتمادية وظيفية (غير بديهية) في هذه الجداول يكون طرفها الأيسر مفتاحاً خارقاً (وهو المفتاح الرئيسي للجدول)؛ غير أن الجدول الأول ليس في الشكل الطبيعي «بويس - كود» لوجود الاعتمادية الوظيفية الجديدة التي تحَدِّد قيمة حقل «رقم المشروع» من قيمة حقل «الموقع» وطرفها الأيسر، «الموقع»، ليس مفتاحاً خارقاً للجدول. ولتطبيع الجدول؛ بحيث يصبح في شكله الطبيعي «بويس - كود»، يمكن تجزئة الجدول إلى جدولين حتى يتوافق مع شروط الشكل الطبيعي «بويس - كود». غير أن عملية تجزئة الجدول ليست بديهية؛ إذ هناك ثلاثة بدائل لعملية التجزئة، وهذه البدائل الثلاثة، كما يلي:

١- (ENO, PNO, Duration, Responsibility) و (ENO, PLocation)

٢- (PNO, ENO, Duration, Responsibility) و (PNO, PLocation)

٣- (PLocation, ENO, Duration, Responsibility) و (PLocation, PNO)

ويوضِّح الشكل رقم (6-7) الشكل العام للاعتمادية الوظيفية التي تخل بشروط الشكل الطبيعي «بويس - كود».



شكل رقم (6–7): الشكل العام للاعتمادية الوظيفية التي تخل بشروط الشكل الطبيعي «بويس – كود».

وتعني الاعتمادية الوظيفية أعلاه وجود مفتاح مرشح (أو رئيسي) يتمثل في الحقلين (A, B) يحَدِّدان قيمة الحقل (C). كما يُوجَد في الوقت نفسه اعتمادية وظيفية هي أن الحقل (C) يحَدِّد قيمة حقل أوَّلي وهو الحقل (B). ونظراً لأن الحقل في الجانب الأيمن من الاعتمادية هو حقل أوَّلي؛ فإن هذه الاعتمادية لا تخالف شروط الشكل الطبيعي الثالث؛ في حين تخالف شروط الشكل الطبيعي «بويس - كود»؛ لكون الحقل في الجانب الأيسر من الاعتمادية وهو الحقل (C) ليس مفتاحاً خارقاً للجدول. ولجعل الجدول في الشكل الطبيعي «بويس - كود»؛ يتمُّ تجزئته إلى جدولين، حسب ما أسلفنا أعلاه، وفق أحد البدائل (أو التوليفات) الثلاثة الممكنة، وهي:

١- ($\underline{A}$, $\underline{B}$) و ($\underline{A}$, $\underline{C}$)

٢- ($\underline{B}$, $\underline{A}$) و (B, $\underline{C}$)

٣- ($\underline{C}$, $\underline{A}$) و ($\underline{C}$, B)

ويُلاحَظ أن كلا الحقلين في الجداول الموجودة في الجهة اليمنى من التوليفات الثلاث أعلاه يمثلان المفتاح الرئيسي للجدول الأول؛ (وذلك إذا كان الحقل (A) والحقل (B) مفتاحاً رئيسياً للجدول الأصلي وليس مفتاحاً مرشحاً). كما يُلاحَظ أن المفتاح الرئيسي للجدول الثاني في البديل الأول يتكوَّن من الحقلين (A) و (C)؛ وذلك لعدم وجود اعتمادية وظيفية بين الاثنين. أمَّا المفتاح الرئيسي للجدول الثاني في كلٍّ من البديل الثاني والبديل الثالث؛ فيتكون من حقلٍ واحد هو الحقل (C)؛ وذلك لأن قيمة هذا الحقل تحَدِّد قيمة الحقل الآخر وفقاً للاعتمادية الوظيفية المخالفة لشروط الشكل الطبيعي «بويس - كود». وفي كلِّ البدائل الثلاثة أعلاه، تدرج بقية الحقول الموجودة في الجدول الأصلي (إن وُجِدَت بالإضافة للحقول الثلاثة المدوَّنة في الجدول أعلاه) ضِمن حقول الجداول الموجودة في الجهة اليمنى من البدائل؛ وذلك لكون بقية الحقول هذه تحتاج إلى مفتاح رئيسي يتكوَّن من حقلين لتحديد قيمهما، كما في المثال السابق.

وفي كلٍّ من التوليفات الثلاث أعلاه؛ يتمُّ فقد الاعتمادية الوظيفية التي تنصُّ على أن الحقل (A) والحقل (B)، مجتمعين، يحَدِّدان قيمة الحقل (C): $A, B \rightarrow C$. أمَّا في مثالنا أعلاه؛ فإنه يتمُّ فقد الاعتمادية الوظيفية ($ENO, PNO \rightarrow PLocation$). وتعني كلمة «فقد»؛ أنه لا يمكن فرض هذه الاعتمادية بعد تجزئة الجدول كما هو الحال قبل تجزئته. وتُسمَّى خاصية المحافظة على الاعتماديات الوظيفية بعد التجزئة بخاصية «المحافظة على الاعتماديات» (DependenCy Preservation). ومن بين البدائل الثلاثة أعلاه، وعلى الرغم من أنها جميعاً ستفقدنا القدرة على فرض قيد الاعتمادية الوظيفية التي أدَّت إلى تجزئة الجدول؛ فإن البديل المقبول الوحيد هو البديل الثالث؛ وذلك لأن بقية البدائل سينتج عنها ما يُسمَّى

«السجلات الزائفة» (Spurious Tuples) عند إجرائنا لعملية «ربط» (Join) بين الجداول الناتجة من كلِّ بديلٍ للحصول على البيانات الأصلية الموجودة في الجدول الأصلي قبل تجزئته. فعلى سبيل المثال: لو استخدمنا البديل الأول في تجزئة جدول
«الموظف - المشروع» سينتج عنه الجدولان التاليان:

EMP_PROJ1_T

ENO	PLocation
E1	Riyadh
E2	Jizan
E2	Jouf
E3	Joubail
E3	Skaka
E4	Jouf
E5	Jouf
E6	Hail
E7	Jizan
E8	Jizan

EMP_PROJ2_T

ENO	PNO	Duration	Responsibility
E1	P1	12	Manager
E2	P3	20	System Analyst
E2	P2	10	System Analyst
E3	P3	10	Consultant
E3	P4	14	Consultant
E4	P2	26	Programmer
E5	P2	14	Manager
E6	P4	16	Programmer
E7	P3	12	Manager
E8	P3	12	Project Secretary

وعند إجرائنا لعملية ربط (طبيعي) بين الجدولين الناتجين من البديل الأول باستخدام الحقل المشترك بين الجدولين وهو حقل «رقم الموظف» (ENO)؛ سينتج أربعة سجلات زائفة؛ لكونها غير موجودة ضمن سجلات الجدول الأصلي (قبل عملية التجزئة). كذلك هو الحال لو استخدمنا البديل الثاني؛ حيث سينتج عن عملية ربط جدولي البديل الثاني سجلات زائفة. ويوضِّح الجدول التالي السجلات الزائفة الناتجة من عملية ربط جدولي البديل الأول موضحة من خلال وضع علامة النجمة (*) أمامها.

EMP_PROJ1_Join_EMP_PROJ2

ENO	PLocation	PNO	Duration	Responsibility	
E1	Riyadh	P1	12	Manager	
E2	Jizan	P3	20	System Analyst	
E2	Jizan	P2	10	System Analyst	*
E2	Jouf	P3	20	System Analyst	*
E2	Jouf	P2	10	System Analyst	
E3	Joubail	P3	10	Consultant	
E3	Joubail	P4	14	Consultant	*
E3	Skaka	P3	10	Consultant	*
E3	Skaka	P4	14	Consultant	
E4	Jouf	P2	26	Programmer	
E5	Jouf	P2	14	Manager	
E6	Hail	P4	16	Programmer	
E7	Jizan	P3	12	Manager	
E8	Jizan	P3	12	Project Secretary	

ونظراً لأن الشكل الطبيعي الثالث والشكل الطبيعي «بويس - كود» ينصَّان على أن شروطهما يجب أن تنطبق على الاعتماديات الوظيفية كافة؛ مما يعني أنه يجب أن تنطبق شروطهما على الاعتماديات الوظيفية الضمنية التي يمكن أن يستدل عليها (أو تستنبط)، بالإضافة إلى الظاهرة منها؛ فإن هذا يدعونا إلى التعرُّف على قواعد الاستدلال. كما أن قواعد الاستدلال هذه ضرورية لمعرفة إن كان أيُّ تجزؤ لجدولٍ ما يحافظ على خاصية «المحافظة على الاعتماديات الوظيفة» (DependenCy Preservation)، وخاصية «السِّجلات غير الزائفة» (Lossless DeComposition).

4-1-6 قواعد الاستدلال (InferenCe Rules):

يقوم مُصمِّمو قواعد البيانات عادةً بتعريف الاعتماديات الوظيفية المتعلقة بكلِّ جدول. وتكون هذه الاعتماديات الوظيفية ذات معانٍ واضحة، ويجب أن تتحقق في أيِّ حالة يكون عليها الجدول، كما رأينا في الأمثلة أعلاه. ويُستخدَم عادةً الرمز "F" للدلالة على هذه الاعتماديات الوظيفية واختصاراً لعبارة «اعتماديات وظيفية» (FunCtional DependenCies). إلا أنه يُوجَد عادةً الكثير من الاعتماديات الوظيفية التي تنطبق على كلِّ حالة من حالات الجدول غير تلك المعرفة في "F". ويمكن الاستدلال على هذه الاعتماديات الوظيفية الأخرى من خلال الاعتماديات الوظيفية المعرفة في "F"؛ إذ إنه يصعب، بشكلٍ عام، تعريف جميع الاعتماديات الوظيفية لتمثيل حالة معينة. فعلى سبيل المثال: لنفترض أن «رقم عضو هيئة التدريس» يحدد «رقم القسم» الذي يعمل فيه عضو هيئة التدريس (Faculty_No $\rightarrow$ Dept_No) وأن «رقم القسم» يحَدِّد «اسم القسم» (Dept_No $\rightarrow$ Dept_Name). بالنظر في هاتين الاعتمادين، معاً؛ نستدل أن «رقم عضو هيئة التدريس» يحَدِّد «اسم القسم» الذي يعمل فيه (Faculty_No $\rightarrow$ Dept_Name). وتُعَدُّ الاعتمادية الثالثة، في هذه الحالة، اعتمادية وظيفية يمكن الاستدلال عليها من خلال الاعتماديتين الوظيفيتين الأخريين، ولا داعي لإدراجها بالإضافة إلى الاعتماديتين الوظيفيتين الأخريين ضِمن الاعتماديات الوظيفية الواجب إيضاحها. وبشكلٍ رسمي؛ يمكن تعريف الاعتماديات الوظيفية كافة التي يمكن أن يستدل عليها من خلال مجموعة الاعتماديات الوظيفية الواضحة "F" فيما يُعرَف بمبدأ «الانغلاق» (Closure)، كما يلي:

> إنَّ مجموعة الاعتماديات الوظيفية التي تحتوي على مجموعة الاعتماديات الوظيفية الواضحة "F"؛ إضافةً لكافة الاعتماديات الوظيفية التي يمكن أن يُستدَل عليها من "F" تُسمَّى انغلاق "F" (Closure of F) ويُرمَز لها بالرمز "F+".

وعلى سبيل المثال: لنفترض تعريف الاعتماديات الوظيفية الواضحة "F" كما يلي:

$$F = \{Faculty_No \rightarrow \{Name, Salary, DOB, Dept_No\},$$
$$Dept_No \rightarrow \{DName, DLocation\}\}$$

فإنه يمكن الاستدلال على اعتماديات وظيفية أخرى من الاعتماديات الوظيفية الواضحة "F"، من ضمنها ما يلي:

$$Faculty_No \rightarrow \{DName, DLocation\}$$
$$Faculty_No \rightarrow Faculty_No$$
$$Dept_No \rightarrow DName$$

وعندما يُستدَل على اعتمادية وظيفية؛ فإن هذه الاعتمادية الوظيفية يجب أن تتحقق على حالات الجدول كافة، كما هو الحال بالنسبة للاعتماديات الوظيفية الواضحة. وللتعرُّف على الاعتماديات الوظيفية كافة التي يمكن أن يستدل عليها من مجموعة من الاعتماديات الوظيفية الواضحة بشكلٍ نمطي؛ فإننا بحاجةٍ إلى قواعد استدلال تمكِّننا من ذلك. ومن أهمِّ قواعد الاستدلال قواعد استدلال «أرمسترونغ» التي تُسمَّى عادة «حقائق أرمسترونغ» (s Axioms'Armstrong) أو «قواعد استدلال أرمسترونغ» (s'Armstrong InferenCe Rules)، وهي كما يلي:

١- الازدياد (Augmentation): $\{X \rightarrow Y\} \Rightarrow \{XZ \rightarrow YZ\}$

٢- الانتقال (Transitivity): $\{X \rightarrow Y, Y \rightarrow Z\} \Rightarrow \{X \rightarrow Z\}$

٣- الانحسار (Reflexivity): $W \subseteq X \Rightarrow \{X \rightarrow W\}$

تنصُّ القاعدة الأولى أنه بإضافة أيِّ مجموعة من الحقول إلى جانبي أيِّ اعتمادية وظيفية؛ تنتج اعتمادية وظيفية صحيحة جديدة. أمَّا القاعدة الثانية فتنصُّ على أن الاعتماديات الوظيفية انتقالية. القاعدة الثالثة تنصُّ على أن أية مجموعة من الحقول تحَدِّد أية مجموعة جزئية من الحقول نفسها. وتُعَدُّ حقائق أرمسترونغ «سليمة» (Sound)؛ بمعنى أنه لا يمكن أن ينتج عنها أية اعتماديات وظيفية خطأ ليست في "F+". كما أن حقائق أرمسترونغ تُعَدُّ «كاملة» (Complete)، بمعنى أنه يمكن لأية مجموعة من الاعتماديات الوظيفية الواضحة "F" الوصول إلى الاعتماديات الوظيفية كافةً في "F+" باستخدام هذه القواعد الثلاث فقط.

وعلى الرغم من أن قواعد أرمسترونغ كاملة؛ فإنه قد تُستخدَم القواعد الإضافية التالية لِتفهُّم بعض الاعتماديات في "F+". وهذه القواعد الإضافية، كما يلي:

١- الاتحاد (Union): $\{X \rightarrow Y, X \rightarrow Z\} \Rightarrow \{X \rightarrow YZ\}$

٢- التفكيك (Decomposition): $\{X \rightarrow YZ\} \Rightarrow \{X \rightarrow Y, X \rightarrow Z\}$

٣- الانتقال الزائف (Pseudotransitivity): $\{X \rightarrow Y, YW \rightarrow Z\} \Rightarrow \{XW \rightarrow Z\}$

ويمكِن الاستدلالُ على جميع الاعتماديات الوظيفية "F+" من الاعتماديات الوظيفية الواضحة "F" بشكلٍ مُبسَّط إذا تمَّ النظر إلى المسألة على أساس معرفة انغلاق مجموعة من الحقول (X+)، وليس معرفة انغلاق مجموعة من الاعتماديات الوظيفية (F+). وتعبر الدالة (FunCtion) التالية عن خوارزمية يمكن من خلالها معرفة انغلاق أيِّ حقل (أو حقول)؛ بحيث تكون مدخلات هذه الدالة مجموعة الحقول المراد معرفة انغلاقها، والممثلة في (X) ومجموعة الاعتماديات الوظيفية المنطبقة على الجدول والممثلة في (F).

```
function ComputeX⁺ (X, F)
begin
X⁺ ← X
While there exists Y → Z ∈ F such that
        Y ⊆ X⁺ and Z ⊈ X⁺
        then X⁺ ← X⁺ ∪ Z
return (X⁺)
end
```

تبدأ الدالة السابقة بوضع جميع الحقول المراد معرفة انغلاقها (X) ضمن انغلاق الحقول (X+) نفسها؛ وذلك حسب قاعدة الانحسار؛ وذلك لأن أية مجموعة من الحقول تحَدِّد قيم أية مجموعة جزئية من الحقول نفسها. بعد ذلك يُنظر في كلِّ اعتمادية وظيفية مطبقة على الجدول (واحدة تلو الأخرى). ولكلِّ اعتمادية وظيفية يُنظر فيما إذا كانت هذه الاعتمادية تمثل اعتمادية انتقالية على الحقول التي تمَّت معرفتها باعتبارها جزءاً من انغلاق (X) وهو (X+). فإذا كانت هذه الاعتمادية هي بالفعل انتقالية وأن حقول جانبها الأيمن لم تُدرَج ضمن (X+) بعد، تُستخدَم قاعدة الانتقال؛ بحيث تتمُّ إضافة هذه الحقول ضمن (X+). ومثال تطبيقي على هذه الدالة؛ لنفترض وجود جدول (R) تنطبق عليه الاعتماديات الوظيفية الظاهرة (F) التالية.

$$F = \{A \rightarrow B, C \rightarrow \{D,E\}, \{E,G\} \rightarrow H\}$$

ولنفترض أننا نرغب في معرفة انغلاق الحقلين C و G. في هذه الحالة تتمُّ مناداة الدالة، كما يلي:

$$ComputeX^{+}(\{C, G\}, F)$$

وبناءً على هذه المعطيات؛ يكون عمل الدالة، كما يلي:

- الوضع المبدئي: $X^+ = \{C, G\}$

- الدورة الأولى: يُنظَر في الاعتمادية الوظيفية التي يكون جزؤها الأيسر مجموعةً من الحقول التي من ضمن +X. يُنظر - إذن - في هذه الحالة إلى الاعتمادية الوظيفية الثانية ($C \rightarrow \{D, E\}$)؛ لكون جانبها الأيسر من ضِمن الحقول الموجودة في +X وحقولها اليمنى ليست من ضِمن حقول +X. ونظراً لأن حقول الجانب الأيمن في الاعتمادية ليست من ضِمن حقول +X على الرغم من أن الحقل الذي يحَدِّد الجانب الأيمن من ضِمن +X، وهو (C)، تتمُّ إضافة الحقول اليمنى من الاعتمادية؛ لتصبح جزءاً من +X. وبذلك تكون نتيجة هذه الدورة كما يلي: {$X^+ = \{C, G, D, E\}$}.

- الدورة الثانية: يُنظر مرةً أخرى في الاعتمادية الوظيفية التي يكون جزؤها الأيسر مجموعةً من الحقول التي من ضمن +X. إذاً يُنظر هنا إلى الاعتمادية الثالثة ($\{E, G\} \rightarrow H$)؛ لكون جانبها الأيسر من ضِمن الحقول الموجودة في +X، وحقولها اليمنى ليست من ضِمن حقول +X. ونظراً لأن حقل الجانب الأيمن في الاعتمادية ليست من ضِمن حقول +X على الرغم من أن الحقول التي تحَدِّد الجانب الأيمن هي من ضِمن +X، تتمُّ إضافة الحقل الأيمن من الاعتمادية ليصبح جزءاً من +X. ويُلاحَظ في هذه الحالة أنه تمَّ استخدام قاعدة الانتقال ضمن خوارزمية الدالة عند تحديد الجانب الأيمن من الاعتمادية الثالثة. وبذلك تكون نتيجة هذه الدورة كما يلي: $X^+ = \{C, G, D, E, H\}$.

- التوقف: يتمُّ التوقف بعد الدورة الثانية لعدم وجود أية اعتماديات وظيفية جانبها الأيسر من ضمن الحقول التي تمَّ التعرف عليها ومن ضمن حقول +X.

وإذا طبقنا هذه الدالة على جدول «الموظف - المشروع» ذي الاعتماديات الوظيفية التالية:

F = {ENO → {ENmae, Title, Salary},

PNO → {PName, Budget},

{ENO, PNO} → {PLocation, Duration, Responsibility},

Title → Salary,

PLocation → PNO}

وذلك بغية معرفة انغلاق بعض حقول الجدول؛ فسيكون انغلاق هذه الحقول، كما يلي:

١. $\{ENO, EName, Title, Salary\} = \{ENO\}^+$
٢. $\{PNO, PName, Budget\} = \{PNO\}^+$
٣. $\{ENO, PNO, EName, Title, Salary, PName, Budget, PLocation, Duration, Responsibility\} = \{ENO, PNO\}^+$
٤. $\{Title, Salary\} = \{Title\}^+$
٥. $\{PLocation, PNO, PName, Budget\} = \{PLocation\}^+$
٦. $\{ENO, PNO, EName, Title, Salary, PName, Budget, PLocation, Duration, Responsibility\} = \{ENO, PLocation\}^+$

ويمكِن تعديل الدالة أعلاه؛ بحيث تُستخدَم للتعرُّف على المفاتيح المرشحة للجدول. ويتم ذلك من خلال معرفة انغلاق كلِّ حقل من حقول الجدول على حِدَة؛ فإذا كانت نتيجة الدالة جميع حقول الجدول يعني أن الحقل مفتاح مرشح. بعد ذلك؛ يتم معرفة انغلاق كلِّ حقلين من حقول الجدول؛ فإذا كانت نتيجة الدالة لأيِّ حقلين جميع حقول الجدول ولم يوجد حقل منها سبق أن كان مفتاحاً مرشحاً للجدول في الخطوة الأولى، يكون الحقلان مفتاحاً مرشحاً للجدول. وتستمر العملية وصولاً إلى حقول الجدول كافة. وتفيد هذه العملية لكون الشكل الطبيعي الثالث والشكل الطبيعي «بويس -كود» ينصان، ضمن شروطهما، على أن يكون الجانب الأيسر من أية اعتمادية وظيفية مفروضة على جدول ما مفتاحاً خارقاً للجدول. ويعني هذا أن الجانب الأيسر يحتوي ضمن حقوله على مفتاح مرشح.

5-1-6 خواص التجزئة (Properties of DeComposition):

يعتمد تطبيعُ الجداول على مبدأ التجزئة للحَدِّ من تكرار البيانات غير المرغوب فيها. وكما أوضحنا عند شرح الشكل الطبيعي «بويس - كود»؛ قد ينتج عن عمليات التجزئة ظهور بعض المشكلات التي لم تكن موجودة أصلاً. وبشكلٍ خاص؛ يجب أن نتأكد من أن تجزئة أيِّ جدول تمكِّننا من استرجاع البيانات الأصلية في الجدول الأصلي عند إجراء عملية ربط بين الجداول الناتجة من عملية التجزئة، كما يجب أن نتأكد من أن تجزئة الجدول تمكِّننا من التحقق من انطباق قيود التكامل، بعد التجزئة، بشكلٍ فعَّال. وفيما يلي شرح للخاصيتين اللتين تمكناننا من اختبار ذلك.

6-1-5-1 خاصية المحافظة على الاعتماديات الوظيفية (Dependency Preservation Decomposition):

إنه من المفيد إذا وُجِدَت كلُّ اعتمادية وظيفية ($X \rightarrow Y$)، موجودة أصلاً ضمن الاعتماديات الوظيفية المفروضة على جدول، وهي "F"، ضِمن أحد الجداول الناتجة من عملية تجزئة الجدول، أو كان بالإمكان الاستدلال عليها من حقول أحد الجداول بعد التجزئة. وتُسمَّى هذه الخاصية «المحافظة على الاعتماديات الوظيفية». وتظهر الحاجة إلى هذه الخاصية لكون كلِّ اعتمادية وظيفية تمثل قيداً بين حقول الجدول الأصلي

يجب التأكد من انطباقها على حالات الجدول كافة. وإذا لم تُوجَد إحدى الاعتماديات الوظيفية ضمن أحد الجداول الناتجة من عملية التجزئة؛ فإننا لن نتمكن من فرض هذه الاعتمادية من خلال التعامل مع جدول واحد؛ ولكنه يجب إجراء عملية ربط بين جدولين أو أكثر، ومن ثم التحقق من انطباق الاعتمادية على نتيجة عملية الربط. ولكون عملية الربط عملية تتطلب بعض الوقت لتنفيذها؛ فإن هذه الطريقة تُعَدُّ غير فعَّالة وغير عملية. وتجدر الإشارة إلى أنه ليس من الضروري أن توجد الاعتماديات الوظيفية الموجودة في "F" ذاتها، كلٌّ على حِدَة، ضِمن أحد الجداول الناتجة من عملية التجزئة؛ لكنه يكفي أن يكون اتحاد الاعتماديات الوظيفية الموجودة في الجداول، كلٌّ على حِدَة، مكافئة للاعتماديات الوظيفية الموجودة في "F". وفيما يلي التعريف الرسمي لخاصية «المحافظة على الاعتماديات الوظيفية»:

تعـد تجزئة هيكل الجـدول R إلى أكثر من جـدول ($R_1, R_2, \ldots, R_n$) تجزئة تحافظ على الاعتماديات الوظيفية إذا كانت كلُّ اعتمادية وظيفية في "F"، بما في ذلك التي يمكن الاسـتدلال عليها، متحققة في اتحاد عمليات إسقاط الاعتماديات الوظيفية على الجداول الناتجة من عملية التجزئة، كما يلي: $$((\pi_{R_1}(F)) \cup (\pi_{R_2}(F)) \cup \ldots \cup (\pi_{R_n}(F)))^+ = F^+$$

ومثالاً على فقد الاعتماديات الوظيفية؛ لننظر في مثال «الموظف - المشروع» الذي تمَّت تجزئته عند شرحنا للشكل الطبيعي «بويس - كود» ونتج عنه ثلاثة بدائل للتجزئة، وهي:

١- (ENO, PNO, Duration, Responsibility) و (ENO, PLocation)

٢- (PNO, ENO, Duration, Responsibility) و (PNO, PLocation)

٣- (PLocation, ENO, Duration, Responsibility) و (PLocation, PNO)

وبإجراء عمليات إسقاط، حسب التعريف أعلاه، للاعتماديات الوظيفية الظاهرة على الجداول الناتجة من البدائل الثلاثة؛ تكون نتيجة عملية الإسقاط، موضَّحة حسب كلِّ بديل، كما يلي:

١- $(\{ENO, PNO\} \rightarrow \{Duration, Responsibility\})$

٢- $((PLocation \rightarrow PNO), (\{PNO, ENO\} \rightarrow \{Duration, Responsibility\}))$

٣- $((PLocation \rightarrow PNO), (\{PLocation, ENO\} \rightarrow \{Duration, Responsibility\}))$

في البديل الأول تمَّ فقد الاعتمادية الوظيفية (PLocation → PNO) وكذلك الاعتمادية الوظيفية ({ENO, PNO} → {PLocation}). وحتى لو تمَّ ربط الجدولين الناتجين من عملية التجزئة في هذا البديل؛ فإننا لن نتمكن من فرض هاتين الاعتماديتين الوظيفيتين؛ وذلك لأن هذا البديل يعاني مشكلةً أكبر وهي وجود سجلات زائفة (Tuples Spurious) كما أوضحنا أعلاه. أمَّا البديل الثاني؛ فإنه يؤدي أيضاً إلى فقد اعتماديات وظيفية، إلا أنه يفقد اعتمادية وظيفية واحدة، وليس اعتماديتين اثنتين كما هو الحال بالنسبة للبديل الأول. وهذه الاعتمادية هي ({ENO, PNO} → {PLocation}). وكما هو الحال بالنسبة للبديل الأول؛ فإن هذا البديل يعاني مشكلةً أكبر لن تمكننا من فرض هذه الاعتمادية؛ لأنه سينتج عن عملية الربط بين جدولي هذا البديل سجلات زائفة. أمَّا البديل الثالث؛ فسيفقد الاعتمادية الوظيفية ({ENO, PNO} → {PLocation}). إلا أن هذا البديل لن ينتج عنه سجلات زائفة في أثناء عملية ربط جدولية، ومن ثم يمكننا التحقق من انطباق الاعتمادية الوظيفية التي فقدت في أثناء التجزئة. ويعني هذا أن كلَّ البدائل الثلاثة لتجزئة الجدول تؤدي إلى فقد اعتمادية وظيفية أو أكثر؛ الأمر الذي يستدعي إجراء عملية ربط بين الجداول الناتجة للتحقق من انطباق الاعتماديات الوظيفية التي فقدت نتيجة لعملية التجزئة، وما دامت عملية التجزئة لا ينتج عنها وجود سجلات زائفة في أثناء ربط الجداول الناتجة من عملية التجزئة. ولهذا السبب؛ فإن خاصية «السجلات غير الزائفة» تُعَدُّ أهم بكثير من خاصية «المحافظة على الاعتماديات الوظيفية»؛ لكون الأولى تُعنَى بصحة البيانات الناتجة بعد عملية الربط، على حين أن الثانية تُعنَى بأداء النظام الذي قد يتأثر نتيجة عمليات الربط؛ بغية التحقق من انطباق الاعتماديات الوظيفية التي فقدت نتيجة لعملية التجزئة.

2-5-1-6 خاصية السجلات غير الزائفة (**Lossless Join DeComposition**):

إن الخاصية الثانية التي يجب لأيِّ تجزئة لجدول أن تتحَلَّى بها؛ هي خاصية السجلات غير الزائفة. وكما أوضحنا عند شرحنا للشكل الطبيعي «بويس - كود»، أنه من الممكن أن تتمَّ تجزئة الجدول وفق أكثر من بديل، ولكن بعض هذه البدائل قد يؤدي إلى إضافة سجلات زائفة عند إجراء عملية ربط بين الجداول الناتجة من التجزئة؛ بغية استعادة البيانات الموجودة في الجدول الأصلي قبل تجزئته. ويعني هذا أن السجلات الزائفة التي قد تتمُّ إضافتها بعد عملية الربط تمثل بيانات خطأ لكونها غير موجودة أصلاً ضمن حالات الجدول قبل تجزئته. وفيما يلي التعريف الرسمي لهذه الخاصية:

تُعَدُّ تجزئة هيكل الجدول R إلى أكثر من جدول $(R_1, R_2, \ldots, R_n)$ تجزئة لا تؤدي إلى وجود سـجلات زائفة عند ربط الجداول الناتجة من عملية التجزئة ربطاً طبيعياً بالنسبة لمجموعـة الاعتماديات الوظيفية "F" المفروضة عـلى R إذا انطبق الشرط التالي على أيّة حالـة "r" من الحالات التي قد يكـون عليها الجدول R، وتنطبق عليه كافة الاعتماديات الوظيفية المفروضة عليه "F":

$$(\pi_{R1}(r) \bowtie \pi_{R2}(r) \bowtie \ \ldots \ \bowtie \pi_{Rn}(r)) = r$$

ويعني التعريف أعلاه أنه لو أخذنا أية حالة "r" يكون عليها الجدول R، وأجرينا عملية إسقاط لحقول الجدول وفق حقول الجداول التي تمثل تجزئة الجدول؛ فإنه يجب أن تكون نتيجة عملية الربط الطبيعي بين جميع جداول التجزئة حالة مكافئة لحالة الجدول الأصلي وهي "r". وبمعنى آخر؛ يجب أن تكون نتيجة ربط جداول التجزئة تحتوي على بيانات تكافئ البيانات الموجودة في الجدول الأصلي دون وجود أيِّ سجلات إضافية.

وللتأكُّد من أن تجزئة أيِّ جدول (R) إلى جدولين (R1, R2) تتحَلَّى بخاصية السجلات غير الزائفة يمكن إجراء الاختبار التالي:

إنَّ تجزئة أية جدول (R) إلى جدولين (R_1, R_2) تتحَلَّى بخاصية السـجلات غير الزائفة بالنسبة للاعتماديات الوظيفية "F" المفروضة على (R) إذا تحقق أيٌّ من الشرطين التاليين:

- تُوجَد اعتمادية وظيفية في "F⁺" على الشكل التالي: $(R_1 \cap R_2) \rightarrow (R_1 - R_2)$
- توجد اعتمادية وظيفية في "F⁺" على الشكل التالي: $(R_1 \cap R_2) \rightarrow (R_2 - R_1)$

ويعني الاختبار السابق أنه عندما نقوم بتجزئة جدول إلى جدولين للحصول على شكل طبيعي ما؛ فإن الجدولين الناتجين يتحليان بخاصية السجلات غير الزائفة إذا كانت الحقول المشتركة بين الجدولين تحَدِّد جميع حقول الجدول الأول أو جميع حقول الجدول الثاني. وبمعنى آخر؛ يجب أن تكون الحقول المشتركة بـين الجدولين الناتجين مفتاحاً رئيسياً للجدول الأول أو مفتاحاً رئيسياً للجدول الثاني. وبناءً على ذلك؛ يمكن إعادة تعريف الشرطين، كما يلي:

- $(R_1 \cap R_2) \rightarrow R_1$
- $(R_1 \cap R_2) \rightarrow R_2$

وبناءً على الاختبار هذا؛ يمكِن التحقق من أن البديل الثالث لتجزئة جدول «الموظف - المشروع» أعلاه هو البديل الوحيد الذي يتحلى بـخاصية السجلات غير الزائفة. ففي هذا الـبديل يوجد حقل واحد مشترك بين الجدولين وهو حقل «موقع المشروع» (PLoCation)، وهذا الحقل يمثل المفتاح الرئيسي لأحد الجدولين. أما في الـبديل الأول فيُوجَد أيضاً حقل مشترك واحد بين جدولي هذا البديل، وهو حقل «رقم الموظف» (ENO)، ولكن هذا الحقل ليس مفتاحاً رئيسياً لأيٍّ من الجدولين. كذلك هو الحال في الـبديل الثاني الذي يُوجَد فيه حقل مشترك واحد بين جدوليه، وهو حقل «رقم المشروع» (PNO)؛ ولكنه ليس مفتاحاً رئيسياً لأيٍّ من جدولي هذا البـديل.

3-5-1-6 التجزئة التي تتحَلَّى بخاصية «المحافظة على الاعتماديات الوظيفية» (DependenCy Preservation))، وخاصية «السجلات غير الزائفة» (Lossless Join DeComposition):

لتطبيع جدول حتى يكون في الشكل الطبيعي الثالث؛ فإنه من الممكن دائماً تجزئته إلى جدولين (أو أكثر)، على أن تتحلى تجزئته بـكلٍّ من خاصية المحافظة على الاعتماديات الوظيفية وخاصية السجلات غير الزائفة. وتمثل الخوارزمية التالية، تعميماً للطريقة التي استخدمناها عند شرح الشكل الطبيعي الثالث لتطبيع أيِّ جدول ليصبح بالشكل الطبيعي الثالث مع المحافظة على كلتا الخاصيتين.

- المدخلات: جدول ليس في الشكل الطبيعي الثالث، والاعتماديات الوظيفية المنطبقة عليه.

- المخرجات: جداول في الشكل الطبيعي الثالث تتحَلى بخاصية «المحافظة على الاعتماديات الوظيفية» وخاصية «السجلات غير الزائفة».

- الخطوات:

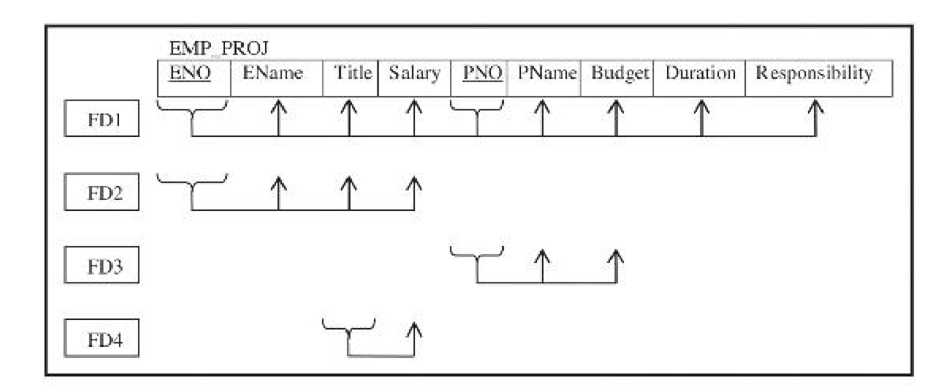
لكافة الحقول الموجودة في الجانب الأيمن من أيَّة اعتمادية وظيفية تحَدِّدها نفس حقول الجانب الأيسر وتكون الاعتمادية الوظيفية مخالفة لشروط الشكل الطبيعي الثالث:

1- يتمُّ إنشاء جدول جديد يحتوي على كافة الحقول الموجودة في الجانب الأيمن التي تحَدِّدها نفس حقول الجانب الأيسر بالإضافة لحقول الجانب الأيسر.

2- يكون مفتاح الجدول الجديد هو الحقول الموجودة في الجانب الأيسر.

3- تزال الحقول التي في الجانب الأيمن من الجدول الأصلي الذي هو قيد التجزئة.

ولإيضاح طريقة عمل الخوارزمية السابقة؛ نعود مرةً أخرى إلى مثال «الموظف - المشروع» المبيَّنة اعتماداته الوظيفية، مرةً أخرى، كما يلي:



وكما سبق أن أوضحنا؛ فإن الاعتمادية الوظيفية الثانية، والثالثة، والرابعة تخالف شروط الشكل الطبيعي الثالث؛ لكونها ليست اعتماديات وظيفية بديهية، وليس الجانب الأيمن فيها حقولاً أولية، كما أن الجانب الأيسر في كلٍّ منها ليس مفتاحاً خارقاً. وبناءً على ذلك يتمُّ إنشاءُ جدول جديد لكلٍّ واحدة من هذه الاعتماديات المخالفة لشروط الشكل الطبيعي الثالث؛ إذ يتمُّ إنشاء جدول يحتوي على جميع الحقول الممثلة في الاعتمادية الثانية، ويكون مفتاح هذا الجدول الحقل الممثل في الجانب الأيسر من الاعتمادية وهو حقل «رقم الموظف» (ENO). وتتمُّ إزالة الحقول الممثلة في الجانب الأيمن من الاعتمادية من حقول الجدول الأصلي. وتكون نتيجة هذه التجزئة، كما يلي:

1- EMP_PROJ (ENO, PNO, PName, Budget, Duration, Responsibility)

2- EMPLOYEE (ENO, EName, Title, Salary)

وينشأ جدول جديد ثانٍ لتمثيل الاعتمادية الوظيفية الثالثة؛ بحيث تكون حقول هذا الجدول مطابقةً لتلك الممثلة في الاعتمادية. وكما هو الحالُ في الجدول السابق، يكون المفتاح الرئيسي للجدول قيد الإنشاء، هو الحقل الممثل في الجانب الأيسر من الاعتمادية، وهو حقل «رقم المشروع» (PNO). وتتمُّ إزالة

الحقول الممثلة في الجانب الأيمن من الاعتمادية من حقول الجدول الأصلي. وتكون نتيجة التجزئة حتى هذه المرحلة كما يلي:

1- EMP_PROJ (ENO, PNO, Duration, Responsibility)

2- EMPLOYEE (ENO, EName, Title, Salary)

3- PROJECT (PNO, PName, Budget)

وبذلك يصبح كلٌّ من الجدول الأصلي (EMP_PROJ) والجدول الثالث (PROJECT) في الشكل الطبيعي الثالث؛ إلا أن الجدول الثاني (EMPLOYEE) ليس في الشكل الطبيعي الثالث؛ لوجود الاعتمادية الوظيفية الرابعة فيه. لذا يتمُّ إنشاء جدول جديد تكون حقوله تلك الممثلة في الاعتمادية الوظيفية الرابعة، ويكون المفتاح الرئيسي للجدول الجديد هو الحقل الموجود في الجهة اليسرى من الاعتمادية، وهو حقل «مُسمى الوظيفة» (Title). وتتمُّ إزالة حقل الجانب الأيمن في الاعتمادية من حقول جدول الموظفين (EMPLOYEE). وبذلك تكون التجزئة النهائية للجدول أربعة جداول كلٌّ منها في الشكل الطبيعي الثالث، كما يلي:

1- EMP_PROJ (ENO, PNO, Duration, Responsibility)

2- EMPLOYEE (ENO, EName, Title)

3- PAYMENT (Title, Salary)

4- PROJECT (PNO, PName, Budget)

4-5-1-6 التجزئة إلى الشكل الطبيعي «بويس - كود» مع المحافظة على خاصية «السجلات غير الزائفة» (Lossless Join DeComposition):

سبق أن أشرنا أعلاه إلى أن هناك تجزئة لأيِّ جدول ليس في الشكل الطبيعي الثالث؛ بحيث تكون نتيجة تجزئته عبارة عن جداول بالشكل الطبيعي الثالث، وتنطبق على هذه التجزئة كلٌّ من خاصية «المحافظة على الاعتماديات الوظيفية» وخاصية «السجلات غير الزائفة». إلا أن هذا من غير الممكن دائماً عند تطبيع جدول ليصبح بالشكل الطبيعي «بويس - كود»؛ بمعنى أن الجداول الناتجة من عملية التجزئة قد لا تنطبق عليها خاصية «المحافظة على الاعتماديات الوظيفية»، كما سبق أن أوضحنا في مثال «الموظف - المشروع» الذي كانت جميع بدائل تجزئته الثلاثة لا تتحلى بخاصية «المحافظة على الاعتماديات

الوظيفية». ولأن خاصية «السجلات غير الزائفة» أهم كثيراً من خاصية «المحافظة على الاعتماديات الوظيفية»، ونظراً لدوام وجود تجزئة لجدول حتى يصبح بالشكل الطبيعي «بويس - كود» مع تحليه بخاصية «السجلات غير الزائفة»؛ فإننا بحاجة إلى طريقة تمكننا من الوصول إلى هذه التجزئة بـشكلٍ فعَّال. وتمثل الخوارزمية التالية هذه الطريقة:

- **المدخلات:** جدول (R) ليس في الشكل الطبيعي «بويس - كود»، والاعتماديات الوظيفية المنطبقة عليه (F).
- **المخرجـات:** مجموعة جداول (D) في الشـكل الطبيعي «بويس - كود» تتحلّى بخاصية «السجلات غير الزائفة».
- **الخطوات:**

١- تتكوّن التجزئة (D) مبدئياً من جدول واحد هو (R).

$$D \leftarrow \{R\}$$

٢- طالمـا وُجِد جدول ضمن جداول التجزئة لا تنطبق عليه شروط الشـكل الطبيعي «بويس - كود» يتمُّ عمل التالي:

- إذا كانت الاعتمادية الوظيفية المخالفة للشكل الطبيعي «بويس - كود» هي:

$$X \rightarrow Y$$

تتمُّ التجزئة كما يلي:

$$D \leftarrow (D-Q) \cup (Q-Y) \cup (X \cup Y)$$

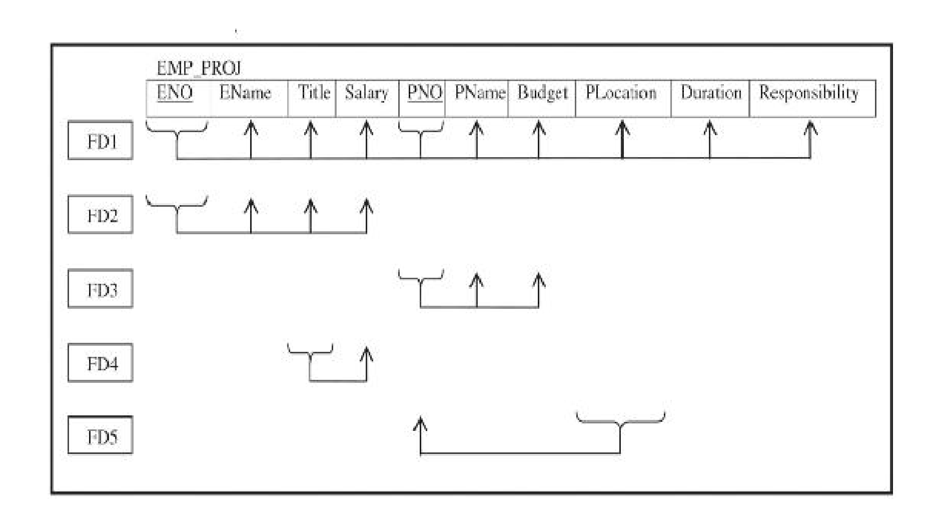
ومثالٌ على طريقة عمل الخوارزمية السابقة؛ نعود مرةً أخرى إلى جدول «الموظف - المشروع» التالي:

EMP_PROJ(ENO, PNO, EName, Title, Salary, PName, Budget, PLocation, Duration, Responsibility)

الذي تنطبق عليه خمس اعتماديات وظيفية وهي كالتالي:

FD1: ({ENO, PNO} → {EName, Title, Salary, PName, Budget, PLocation, Duration, Responsibility}
FD2: ENO → {EName, Title, Salary}
FD3: PNO → {PName, Budget}
FD4: Title → Salary
FD5: PLocation → PNO

وتمثل هذه الاعتماديات الوظيفية المطبقة على الجدول بالرسم كما يلي:



وكما أسلفنا سابقاً؛ إن جدول «الموظف - المشروع» أعلاه ليس في الشكل الطبيعي «بويس – كود»؛ وذلك بسبب الاعتماديات الوظيفية الثانية والثالثة والرابعة والخامسة؛ إذ إن الجانب الأيسر في كلٍّ منها ليس مفتاحاً خارقاً للجدول. ولتطبيع الجدول ليصبح في الشكل الطبيعي «بويس - كود» نستخدم الخوارزمية السابقة، بافتراض أن اسم الجدول هو (1R)، كما يلي:

- الوضع المبدئي: تتكون التجزئة من جدولٍ واحد حقوله هي جميع حقول الجدول الرئيسي: .D= {R1}

- الدورة الأولى: R1 الموجود ضِمن جداول التجزئة ليس في الشكل الطبيعي «بويس -كود». يتم اختيار إحدى الاعتماديات الوظيفية المنطبقة على R1، وفي الوقت نفسه مخالفة للشكل الطبيعي «بويس - كود»، ولتكن:

$$ENO \rightarrow \{EName, Title, Salary\}$$

تطبق قاعدة التجزئة . ولأن (D) في الدورة الأولى تحتوي على جدول واحد؛ فهو أيضاً (Q). وتكون نتيجة تطبيق القاعدة، كما يلي:

$$D \leftarrow \emptyset \cup R_2 \cup R_3$$

بحيث إن:

R2 (ENO, PNO, PName, Budget, PLoCation, Duration, Responsibility)

R3 (ENO, EName, Title, Salary)

- الدورة الثانية: تكون جداول التجزئة حتى الآن كما يلي: {R3 ,R2} =D.

الجدول 2R والجدول 3R ليسا في الشكل الطبيعي «بويس - كود». تتمُّ تجزئة الجدول 3R وفق الاعتمادية الوظيفية المنطبقة على 3R وفي الوقت نفسه مخالفة للشكل الطبيعي «بويس - كود»، وهي:

$$Title \rightarrow Salary$$

تطبق قاعدة التجزئة $D \leftarrow (D\text{-}Q) \cup (Q\text{-}Y) \cup (X \cup Y)$ على الجدولR3، كما يلي:

$$D \leftarrow (R_2) \cup (R_4) \cup (R_5)$$

بحيث إن:

R_4 (<u>ENO</u>, EName, Title)

R_5 (<u>Title</u>, Salary)

- الدورة الثالثة: تكون جداول التجزئة حتى الآن كما يلي: {R2, R4, R5,} =D.

الجدول R4 والجدول R5 في الشكل الطبيعي «بويس - كود»؛ ولكن الجدول R2، ليس كذلك. يتمُّ اختيار إحدى الاعتماديات الوظيفية المنطبقة على R2، وفي الوقت نفسه مخالفة للشكل الطبيعي «بويس - كود»، ولتكن:

$$PNO \rightarrow \{PName, Budget\}$$

تطبق قاعدة التجزئة $D \leftarrow (D\text{-}Q) \cup (Q\text{-}Y) \cup (X \cup Y)$ على الجدول R2، كما يلي:

$$D \leftarrow (R_4 \cup R_5) \cup (R_6) \cup (R_7)$$

بحيث إن:

R_6 (<u>ENO</u>, <u>PNO</u>, PLocation, Duration, Responsibility)

R_7 (<u>PNO</u>, PName, Budget)

- الدورة الرابعة: تكون جداول التجزئة حتى الآن كما يلي: {R4, R5, R6, R7,} =D.

الجدول R4 والجدول R5 والجدول R7 في الشكل الطبيعى «بويس – كود»؛ ولكن الجدول R6 ليس كذلك. تتمُّ تجزئة الجدول وفق الاعتمادية الوظيفية المنطبقة على R6 وفي الوقت نفسه مخالفة للشكل الطبيعي «بويس - كود»، وهي:

$$PLocation \rightarrow PNO$$

تطبق قاعدة التجزئة $D \leftarrow (D\text{-}Q) \cup (Q\text{-}Y) \cup (XUY)$ على الجدولR6، كما يلي:

$$D \leftarrow (R_4 \cup R_5 \cup R_7) \cup (R_8) \cup (R_9)$$

بحيث إن:

R_8 (<u>ENO</u>, <u>PLocation</u>, Duration, Responsibility)

R_9 (<u>PLocation</u>, PNO)

ويُلاحَظ في التجزئة الأخيرة أعلاه أن حقل «موقع المشروع» (PLoCation)؛ قد أصبح جزءاً من المفتاح الرئيسي للجدول الجديد (R8) عوضاً عن حقل «رقم المشروع» (PNO) الذي تم نقله للجدول الجديد (R9) حتى تصبح الحقول التي تمثل تقاطع الجدولين (وهو الحقل «رقم المشروع» (PLoCation)) مفتاحاً رئيسياً لأحدهما كما تنص عليه قاعدة التجزئة التي تتحلى بخاصية السجلات غير الزائفة، وخاصةً أنه يمكن تحديد رقم المشروع من خلال موقعه حسب الاعتمادية الوظيفية التي أدَّت إلى تجزئة الجدول.

- **المخرجات:** يتمُّ التوقف لعدم وجود أي جدول مخالف للشكل الطبيعي «بويس - كود» ضمن مجموعة جداول التجزئة D. وتكون النتيجة النهائية لمجموعة الجداول، كما يلي:

D = { R_4(<u>ENO</u>, EName, Title),
R_5(<u>Title</u>, Salary),
R_7(<u>PNO</u>, PName, Budget),
R_8(<u>ENO</u>, <u>PLocation</u>, Duration, Responsibility),
R_9(<u>PLocation</u>, PNO)
}

6-1-6 الشكل الطبيعي الرابع (Fourth Normal Form (4NF)):

عندما يكون جدول ما في الشكل الطبيعي «بويس - كود»؛ فإن مثل هذا الجدول لن يحتوي على أيٍّ من مشكلات التعديل، التي سبق أن أوضحناها أعلاه؛ بسبب الاعتماديات الوظيفية. إلا أن مثل هذا الجدول قد يحتوي على مشكلات أخرى نتيجة لما يُعرَف بـ «الاعتماديات متعددة القيم» (Multivalued DependenCies (MVD)). فعلى سبيل المثال، لنفترض وجود جدول بمسمى «المواد الدراسية»، كما يلي:

COURSE_T

Course_ID	Instructor_Name	Textbook
Physics	Alhamad	Introduction to Physics
Physics	Albandar	Physics Fundamentals
Physics	Albandar	Understanding Physics
Physics	Albandar	Introduction to Physics
Chemistry	Elmasri	Introduction to Chemistry
Chemistry	Elmasri	Basic Chemistry
Mathematics	Albader	Basic Mathematics

إن الجدول أعلاه يتكون من ثلاثة حقول تمثل المادة الدراسية، وهي: رمز المادة الدراسية (Course_ID)، واسم عضو هيئة التدريس الذي يقوم بتدريس المادة الدراسية (InstruCtor_Name)، والمرجع العلمي المستخدم في تدريس المادة (Textbook). ولأن الحقول الثلاثة التي يتكون منها الجدول تمثل مفتاحه الرئيسي؛ فإن الجدول بالشكل الطبيعي «بويس - كود». ومع أنه لا توجد في الجدول أيٌّ اعتماديات وظيفية تخالف شروط الشكل الطبيعي «بويس - كود»، وأن الاعتمادية الوظيفية (الظاهرة) الوحيدة هي تلك المتعلقة بالمفتاح الرئيسي، كما يلي:

{Course_ID, Instructor_Name, Textbook} → {Course_ID, Instructor_Name, Textbook}

إلا أن الجدول يعاني مشكلات التعديل، كما يلي:

1- مشكلة الحذف: لو تم حذف عضو هيئة التدريس الذي اسمه "Albader"؛ فإننا سنفقد أيضاً مسمى المرجع الدراسي المستخدم في تدريس مادة الرياضيات (MathematiCs).

2- مشكلة الإضافة: لو أردنا إضافة مرجع إضافي لمادة الفيزياء (PhysiCs)؛ فإننا سنضطر إلى إضافة ثلاثة سجلات؛ بحيث يضاف سجل جديد لكلِّ عضو من أعضاء هيئة التدريس الثلاثة الذين يقومون بتدريس مادة الفيزياء.

3- مشكلة التحديث: لو أردنا تحديث مسمى أحد مراجع مادة الفيزياء؛ فإننا سنضطر لإجراء عملية التحديث في ثلاثة سجلات. فعلى سبيل المثال: لو أردنا تحديث المرجع "IntroduCtion to PhysiCs" ليصبح في طبعته الثانية "(2ed) IntroduCtion to PhysiCs". فإنه يجب تغيير هذا المسمى في ثلاثة سجلات؛ وذلك لكل عضو من أعضاء هيئة التدريس الذين يقومون بتدريس مادة الفيزياء.

وتظهر الاعتماديات متعددة القيم عندما يكون في الجدول ثلاثة حقول على الأقل؛ بحيث إن إحدى هذه الحقول وليكن «حقل1» يحدد مجموعة محددة من قيم حقلٍ ثانٍ، وليكن «حقل2»، كما أن «حقل1» يحدد مجموعةً محددةً من حقل ثالث، وليكن «حقل3» وأن مجموعة قيم «حقل2» و«حقل3» مستقلتان. ففي المثال أعلاه، يحدد رمز المادة الدراسية مجموعةً محددةً من قيم أسماء أعضاء هيئة التدريس، وهي لأولئك الذين يقومون بتدريس المادة الدراسية، كما أنه يحَدِّد مجموعة محددة من المراجع العلمية المرتبطة بكل مادة دراسية، وأن قيم مجموعة أسماء أعضاء هيئة التدريس الذين يقومون بتدريس مادة دراسية مستقلة عن قيم المراجع العلمية الخاصة بالمادة الدراسية. وتظهر مثل هذه الاعتماديات المتعددة القيم في الغالب عندما يتكون الجدول من مجموعة من الحقول تكون في مجملها مفتاحاً رئيسياً للجدول.

ولتطبيع أيِّ جدول؛ بحيث يكون في الشكل الطبيعي الرابع؛ فإنه يجب أن يكون الجدول في الشكل الطبيعي «بويس - كود»، وفي الوقت نفسه، لا يحتوي على أي اعتماديات متعددة القيم. ويمكن النظر إلى الجدول السابق، كما يلي:

COURSE

Course_ID	Instructor_Name	Textbook
Physics	Alsaleh Alhamad Albandar	Physics Fundamentals Understanding Physics Introduction to Physics
Chemistry	Elmasri	Introduction to Chemistry Basic Chemistry
Mathematics	Albader	Basic Mathematics

يوضِّح الجدول السابق كما لو كان الجدول الأصلي مكوناً من ثلاثة حقول؛ اثنان منها (وهما: حقل اسم عضو هيئة التدريس، وحقل المرجع العلمي) حقلان متعددا القيم، وأن الحقل الذي يحدد مجموعة القيم التي من الممكن أن يأخذها أيٌّ من الحقلين هو حقل رمز المادة الدراسية. كما يوضِّح الجدول أن مجموعة

قيم كلا الحقلين متعددي القيم منفصلة عن بعضها. ولتحويل الجدول إلى الشكل الطبيعي الرابع؛ تتم تجزئته إلى جدولين؛ بحيث يحتوي الجدول الأول على أسماء أعضاء هيئة التدريس الذين يقومون بتدريس المواد الدراسية بمسمَّى «المؤهلات التدريسية» (QUALIFICATION) في حين يحتوي الثاني على المراجع العلمية الخاصة بالمواد الدراسية بمسمى «المواد المرجعية» (REFERENCE_MATERIAL)، كما يلي:

QUALIFICATION

Course_ID	Instructor_Name
Physics	Alsaleh
Physics	Alhamad
Physics	Albandar
Chemistry	Elmasri
Mathematics	Albader

REFERENCE_MATERIAL

Course_ID	Textbook
Physics	Physics Fundamentals
Physics	Understanding Physics
Physics	Introduction to Physics
Chemistry	Introduction to Chemistry
Chemistry	Basic Chemistry
Mathematics	Basic Mathematics

ويُلاحَظ في طريقة تجزئة الجدول السابقة، التي تمَّ فيها إنشاء جدول جديد لكل مجموعة متعددة القيم؛ أنها لم تقم بالتغلب على مشكلات التعديل فحسب، وإنما قد قلصت عدد السجلات؛ إذ إن الجدولين الناتجين يحتويان على أحد عشر (11) سجلاً، في حين يحتوي الجدول الأصلي على اثني عشر (12) سجلاً. ويعني هذا أن تحويل أي جدول إلى الشكل الطبيعي الرابع يسهم في تقليص حجم المساحة التخزينية المطلوبة لتخزين الجدول، وخاصةً عندما تكون مجموعة القيم التي من الممكن أن تكون عليها الحقول متعددة القيم كثيرة جداً.

7-1-6 الأشكال الطبيعية العليا (Higher Normal Forms):

تُوجَد أشكالٌ طبيعية أخرى ذات خصائص تختلف عما تمَّ ذكره حتى الآن، مثل الشكل الطبيعي الخامس (5NF)؛ ولكن الحاجة الفعلية التي تتطلب تطبيع الجداول حتى تصبح بهذه الأشكال الطبيعية العليا من النادر جداً أن تظهر على أرض الواقع. لذا؛ فإن هذه الأشكال الطبيعية تُعَد ذات قيمة نظرية أكثر من كونها ذات قيمة تطبيقية. ولأن محتويات هذا الكتاب تميل إلى أن تكون ذات صبغة تطبيقية؛ فإننا لن نقوم باستعراض وشرح هذه الأشكال الطبيعية العليا.

2-6 التصميم المادي لقواعد البيانات العلاقية:

إن الهدف من التصميم المادي لنظم قواعد البيانات؛ هو إنشاء تصميم يُمكِّن من تخزين البيانات بشكلٍ يوفر الأداء المناسب لنظام إدارة قاعدة البيانات على اختلاف حجم العمليات التي تنفذ على قاعدة البيانات. ويعني هذا؛ وعلى خلاف التصميم المفاهيمي والتصميم المنطقي، أن التصميم المادي يوضح الكيفية التي ستُخزَّن وتُعالَج فيها البيانات لا على الكيفية التي يتمُّ من خلالها التعرف على البيانات والعلاقات فيما بينها أو طريقة تمثيلها وفق النموذج العلاقي أو نماذج البيانات الأخرى. وبناءً على هذا؛ يوضِّح هذا الجزء من الكتاب خيارات تخزين قيم الحقول وكيفية تحديد الخيار المناسب. ويوضِّح هذا الجزء أيضاً أن الجداول التي تمَّ تطبيعها (أو المطبعة) قد لا تكون الشكل الأمثل الذي من المفترض أن تُخزَّن فيه في ملفات قاعدة البيانات؛ مما قد يستدعي فكَّ التطبيع لتحسين أداء نظام قاعدة البيانات. كما يقارن هذا الجزء بين تنظيمات الملفات وعلى طرق استخدام الفهارس التي تُسهِم في تسريع استرجاع البيانات من قاعدة البيانات. ويجب في أثناء التصميم المادي لقاعدة البيانات توخي الحذر الشديد؛ إذ إن الخيارات التي يتمُّ اتخاذها في هذه المرحلة تؤثر في فاعلية الوصول للبيانات، وأمن البيانات، وسهولة التعامل مع قاعدة البيانات من قِبل المستفيدين.

6–2–1 عملية التصميم المادي:

إن الهدف الرئيسي من عملية التصميم المادي لقواعد البيانات؛ هو التمكُّن من معالجة البيانات بشكلٍ فعَّال؛ وذلك من خلال تقليص الوقت اللازم الذي تحتاج إليه التعليمات الصادرة من قِبل المستفيدين (وبرامج التطبيقات) للتفاعل مع البيانات المخزَّنة في قاعدة البيانات. وعلى الرغم من أن المساحة التخزينية المُستخدَمة لتخزين قاعدة البيانات تُعَدُّ مهمةً في بعض الأحيان؛ فإنها بدأت تفقد أهميتها في السنوات الأخيرة؛ وذلك للتقليل المستمر من تكلفة المساحات التخزينية للحاسبات الآلية. لذلك؛ فإن مرحلة التصميم المادي تركِّز عادة، وبشكل أكبر، على التسريع في عملية التعامل مع الملفات التي تستخدمها قواعد البيانات، وعلى البيانات المخزَّنة في قاعدة البيانات دون تركيز كبير على الفاعلية في استخدام المساحة التخزينية المتاحة.

وتتطلب عملية التصميم المادي للملفات، وقواعد البيانات بعض المعلومات التي تساعد على التصميم المادي الجيد، ومن المفترض أن يكون قد تمَّ جمعها في أثناء مراحل سابقة لتصميم النظام المعلوماتي. ومن هذه المعلومات، ما يلي (Hoffer et al, 2015):

- علاقات (أو جداول) قاعدة البيانات بعد تطبيعها مع تقدير أحجام هذه العلاقات، أيْ: تقدير عدد السجلات في جداول قاعدة البيانات.

- تعريف لكلِّ حقل من حقول قاعدة البيانات.

- توصيف للتوقيت والمكان اللذين يحَدِّدان متى وأين يتمُّ التعامل مع البيانات، سواء من خلال عمليات الإدخال أو الاسترجاع أو الحذف أو التحديث مع تحديد لأعداد هذه العمليات على البيانات.

- التوقعات أو المتطلبات المتعلقة بالوقت اللازم للنظام للتعامل مع التعليمات المصدَّرة إليه (Response Time)، وبأمن البيانات، وبالتخزين الاحتياطي، وبالوقت اللازم للاستعادة بعد حدوث الأعطال، وبتناسق (أو تكامل) البيانات.

- توصيف للتقنيات المُستخدَمة في بناء قاعدة البيانات متضمناً ذلك نظام التشغيل ونظام إدارة قاعدة البيانات.

وتحتاجُ عملية التصميم المادي إلى اتخاذ عددٍ من القرارات المهمة التي تؤثر في تكامل وأداء نظم التطبيقات التي ستتعامل مع قاعدة البيانات. ومن أهم هذه القرارات، ما يلي:

- اختيار هيئة البيانات (Data Format) التي ستُخزَّن عليها بيانات كل حقل من حقول قاعدة البيانات التي وردت في التصميم المنطقي لقاعدة البيانات. ويتمُّ اختيار هيئة البيانات بأقل ما يمكن من مساحة تخزينية وبأكثر ما يمكن من تكامل للبيانات.

- تجميع حقول البيانات الواردة في التصميم المنطقي على هيئة سجلات مادية. وعلى الرغم من أن تجميع الحقول حسب ورودها ضمن أعمدة الجداول التي تمَّ تصميمها في مرحلة التصميم المنطقي لقاعدة البيانات ضِمن سجلات مادية قد يبدو أمراً منطقياً إلا أنه ليس دائماً الشكل الأمثل لتجميع الحقول بشكلٍ مباشرٍ ضمن السجلات المادية.

- ترتيب السجلات ذات الهياكل المتشابهة في الذاكرة الثانوية بشكلٍ يساعد على تخزينها، وتحديثها، واسترجاعها بشكلٍ سريع سواء كان تنفيذ هذه العمليات على سجلات منفردة أم على مجموعات منها.

- اختيار فهارس تساعد على تخزين وربط ملفات قاعدة البيانات بشكلٍ يجعل استرجاع البيانات أكثر فاعلية.

- وَضْع إستراتيجيات للتعامل مع الاستفسارات التي تُنفَّذ على قاعدة البيانات؛ بحيث تحسِّن أداء تنفيذ هذه الاستفسارات وتستفيد من تنظيم ملفات قاعدة البيانات والفهارس التي تم تعريفها.

6-2-1-1 تصميم الحقول:

يُعَدُّ الحقل أصغر وحدة تخزين للبيانات التي يمكِن التعامل معها من قِبل لغات البرمجة، أو نظم إدارة قواعد البـيانات. ومن أهمِّ الأمور التي تجب مراعاتها في أثناء تصميم حقول البيانات التي ستتضمنها قاعدة البيانات نوعية البيانات (Data Type)، والتحكم في تناسق البيانات (Data Integrity) التي ستحتويها الحقول، والكيفية التي سيقوم من خلالها نظام إدارة قاعدة البيانات بالتعامل مع القيم غير المدخلة (أو المفقودة) (Missing Data).

إنَّ اختيار نوعية البيانات المناسبة لحقول قاعدة البيانات له أربعة أهداف رئيسية تختلف أهميتها النسبية باختلاف التطبيقات التي تتعامل معها، وهي:

- تقليص المساحة التخزينية المُستخدَمة من قِبل الحقل.

- التمكُّن من تمثيل جميع القيم التي من الممكن أن يحتوي عليها الحقل.

- القدرة على تحسين تكامل البيانات.

- التمكُّن من دعم جميع العمليات التي قد تُجرى على القيم المخزَّنة في الحقل.

واختيار نوعية البيانات المناسبة لحقل ما يُمكِّن من تمثيل كلِّ القيم التي من الممكن أن يأخذها الحقل الذي يقابله في النموذج المفاهيمي، وبأقل قدرٍ ممكن من المساحة التخزينية، مع عدم إمكانية تمثيله للقيم غير المسموح بها (أيْ: تحسين تكامل البـيانات). كما أن اختيار نوعية الـبيانات المناسبة للحقل يمكِّن من دعم العمليات التي قد تُجرَى على القيم المخزنة في الحقل مثل إجراء العمليات الحسابية إذا كان الحقل مُمثِّلاً لبيانات عددية أو إجراء العمليات الخاصة بالحروف (String Manipulation) إذا كان الحقل مُمثِّلاً لبـيانات حرفية.

وفي العديد من نظم إدارة قواعد البيانات يُمكِن التحكم في تكامل البيانات من خلال وضع قيود على الحقول ضمن هياكل الحقول نفسها، ومن ثم فرض هذه القيود على الحقول من قِبل نظم إدارة قواعد البـيانات. وتحَدِّد نوعية الـبيانات التي يتمُّ اختيارها لحقل ما من فرض نوعٍ واحدٍ من قيود تكامل البيانات؛ بحيث تكون بيانات الحقل حرفية أو عددية (بـالإضافة لطول الحقل). إلا أن هنالك أنواعاً أخرى من القيود التي يمكن فرضها على الحقول، ومن أكثرها شيوعاً، ما يلي:

- القيمة الافتراضية (Default Value): هي القيمة التي سيأخذها الحقل عند عدم إدخال قيمة له في أثناء إدخال سجل جديد. وتمكِّن القيم الافتراضية للحقول من إدخال الـبيانات بشكلٍ سريع من قِبل المستفيدين؛ وذلك عندما تكون الغالبية العظمى من السجلات التي يتمُّ إدخالها لها قيمة تساوي القيمة

الافتراضية للحقل، ومن ثم يمكن تخطي عملية إدخال قيم لحقول السجلات التي تساوي القيمة الافتراضية للحقل؛ مما يسهم في الإسراع في عملية إدخال البيانات.

- التحكُّم في مدى القيم (Domain Values): يمكن فرض قيود على مجموعة القيم التي من الممكن أن تأخذها قيم حقلٍ ما. ويمكن أن يكون المدى عددياً يتمُّ تحديده بقيمة دنيا وقيمة عليا أو مجموعة من القِيم المحَدَّدة سواء أكانت عدديةً أو سلاسل حرفية أو تواريخ.

- التحكُّم في القِيم الغائبة (Null Values): قد يُسْمَح لبعض الحقول أن تكون قيمها غائبةً، في حين لا يُسْمَح لحقول أخرى بذلك. ويساعد التحكم في القيم الغائبة للحقول على تكامل البيانات. فعلى سبيل المثال: قد تمنع السياسة العامة لإحدى الجامعات إضافة أية مادة دراسية جديدة ما لم يكن للمادة الدراسية اسم (مثل: مقدمة في الحاسب الآلي أو نظم قواعد البيانات). كذلك هو الحال بالنسبة لسجلات الموظفين؛ فقد تمنع السياسة العامة لمنظمة ما من إضافة أيِّ سجل لموظف جديد ما لم يتم إدخال قيمة في حقل رقم الموظف.

- السلامة المرجعية (Referential Integrity): تُعَدُّ السلامة المرجعية أحد أشكال التحكم في المدى؛ إذ إن القيم التي بإمكان الحقل أن يأخذها يجب أن تكون من ضِمن قيم حقل آخر موجود في سجل آخر ضِمن الجدول نفسه أو جدول آخر. ويعني هذا أن مدى القيم المفروض على حقل من هذا النوع؛ ذو طبيعة ديناميكية تختلف مع مرور الزمن باختلاف القيم التي يحتويها الحقل الآخر الذي تُسْتَمَدُّ منه هذه القيم عوضاً عن كون مدى قيم الحقل ثابتة ومعرفة بشكل مسبق.

2-1-2-6 تصميم السِّجلات، وعملية فكِّ التطبيع:

في أثناء عملية التصميم المنطقي؛ يتمُّ تجميع الحقول ضِمن صفوف في جداول قاعدة البيانات؛ بحيث يمثل كلُّ صف حالة معينة يتمُّ تحديدها من خلال المفتاح الرئيسي للجدول الذي يحتوي على الصف. وعلى النقيض من ذلك؛ فإن التصميم المادي للحقول يعني وضعها بشكلٍ يجاور بعضها بعضاً في الذاكرة الثانوية؛ بحيث يمكِن استرجاعها وكتابتها كوحدات (Pages) من قِبل نظام إدارة قاعدة البيانات. لذلك؛ فإن التصميم المادي للحقول يتطلب اختيار ترتيب الحقول بشكلٍ متجاور، حتى يمكن: (1) الاستفادة من الذاكرة الثانوية بشكلٍ فعَّال، و(2) معالجة البيانات بشكلٍ سريع.

وتتأثر درجة الاستفادة من الذاكرة الثانوية بشكلٍ كبيرٍ بمقاس الحقول وهيكلية الذاكرة الثانوية؛ فنظم التشغيل تتعامل مع البيانات المخزَّنة في الذاكرة الثانوية (سواء بقراءتها أو كتابتها) كوحدات تُسمَّى صفحات (Pages). والصفحة تمثل كمية البيانات التي بإمكان نظام التشغيل قراءتها أو كتابتها من خلال عملية

إدخال (للذاكرة الرئيسية)، أو إخراج (من الذاكرة الرئيسية) واحدة. ومقاس الصفحة مقاس ثابت يتمُّ تحديده من قِبل مديري قواعد البيانات؛ بحيث تتمُّ الاستفادة القصوى من حجم الذاكرة الرئيسية لجهاز الحاسب من قِبل جميع التطبيقات التي تنفذ عليه. وبحسب طبيعة الحاسب الآلي المُستخدَم؛ فإنه قد يسمح بتوزيع محتويات السجل الواحد على صفحتين أو قد لا يسمح بذلك. ويعني هذا أنه قد يتمُّ إهدار الكثير من المساحة التخزينية في حالِ كان مقاس الصفحة الواحدة لا يمثل حاصل ضرب طول السجل الواحد بعددٍ صحيح؛ مما يترك بعض المساحة التخزينية المهدرة في نهاية كلِّ صفحة. ويُسمَّى عدد السجلات التي بالإمكان تخزينها في الصفحة الواحدة بمعامل الكتلة (BloCk FaCtor (BF)). وعندما تكون المساحة التخزينية المتوافرة في الحاسب الآلي قليلةً (نسبياً)، وفي الوقت نفسه لا يمكن أن توزع محتويات السجلات على أكثر من صفحة؛ فإن إنشاء أكثر من ملف لجدول ما وتوزيع حقول سجلات الجدول عليها يقلِّص من المساحة التخزينية المُهدَرة.

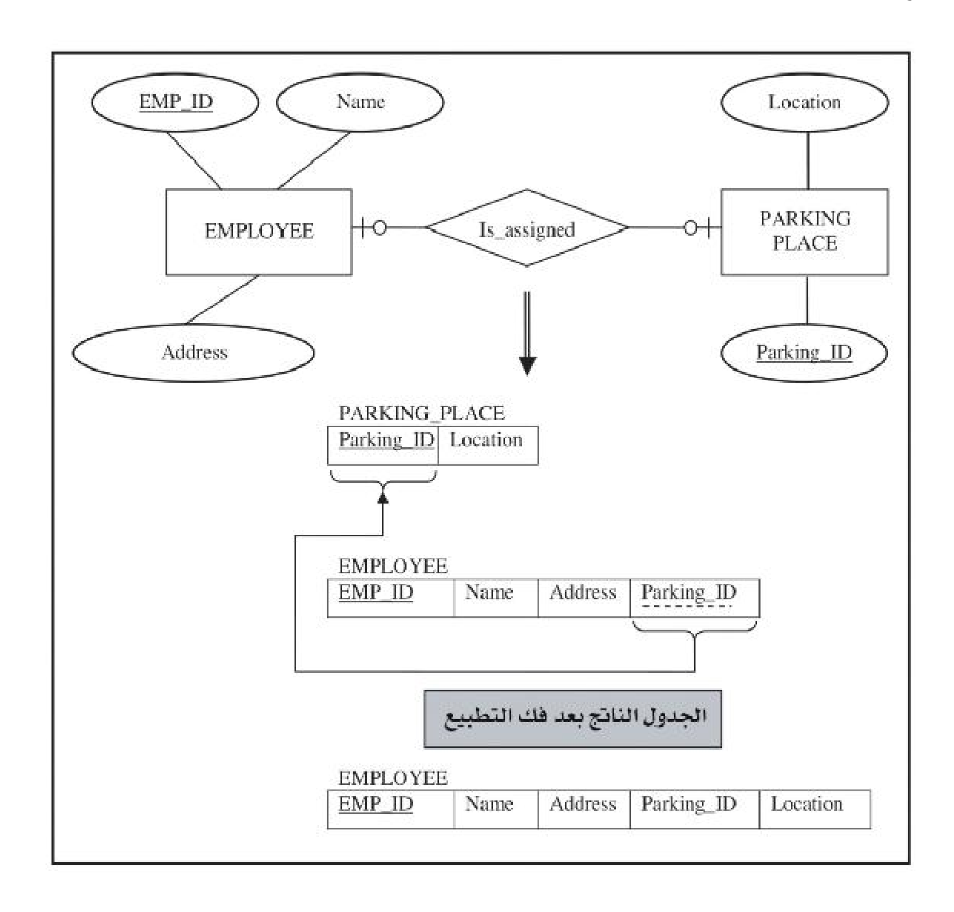
على الرَّغم من أن تقليص المساحة التخزينية المُهدَرة يُعَدُّ أحد العوامل المهمة في عملية التصميم المادي لقاعدة البيانات؛ فإن عامل التسريع في معالجة البيانات يفوق في أهميته العامل الآخر. وتعتمد سرعة معالجة البيانات على مدى قرب البيانات المرتبطة بعضها ببعض (والتي تتعامل معها التعليمات الواردة للنظام بشكلٍ متكرر) في وسائل التخزين الثانوية. ففي العادة لا يتم استخدام حقول الجداول كافة؛ للإجابة عن العمليات التي تتفاعل مع قاعدة البيانات، وإنما يتمُّ استخدام حقول بيانات تابعة لأكثر من جدول عند تنفيذ العمليات التي ترد لنظام إدارة قاعدة البيانات. وبناءً على ذلك؛ فإن الجداول التي تمَّ تطبيعها في أثناء عملية التصميم المنطقي لقاعدة البيانات التي تقلل من تكرارية البيانات، ومن ثم الحَدّ من أخطاء التعديل قد لا تكون الشكل الأمثل الذي يؤدي إلى معالجة فعَّالة للبيانات إذا ما تم بناؤها مادياً بالشكل نفسه الذي صُمِّمت عليه منطقياً. وقد أوضحت بعض الدراسات أن الفاعلية في معالجة بيانات قاعدة بيانات مبنية مادياً حسب بنائها المنطقي المطبع بشكلٍ كاملٍ قد يكون مكلفاً وبشكلٍ كبيرٍ مقارنةً ببناء قاعدة البيانات نفسها؛ ولكن من خلال تطبيعها بشكلٍ جزئي (Inmon,1988). وعلى الرغم من أن نتائج مثل هذه الدراسات تعتمد على محتويات قاعدة البيانات وطبيعة العمليات التي تنفذ عليها؛ فإنَّ مثل هذه الدراسات توضِّح ضرورة توخِّي الحرص عند بناء قاعدة البيانات مادياً من حيث بنائها بشكلٍ يتوافق مع تطبيعها بشكلٍ كاملٍ، أو بنائها بشكلٍ أقل درجة في التطبيع مقابل تخفيض تكلفة معالجة التعليمات التي ترد لنظام إدارة قاعدة البيانات.

وتهدف عملية فك التطبيع (Denormalization) إلى تحويل العلاقات (أو الجداول) المطبعة إلى سجلات يتمُّ تخزينها مادياً بشكلٍ أقل تطبيعاً. وقد ينتجُ عن عملية فكِّ التطبيع تجزئة حقول كلِّ سجل من سجلات علاقة ما إلى مجموعة من السجلات عوضاً عن تخزينها مادياً كسجل واحد، أو أن يتمَّ دمجُ حقول سجلات تابعة لأكثر من علاقة في سجل مادي واحد أو كلا الاثنين معاً. وعلى الرغم من وجود الكثير من

السلبيات التي قد تنتج عن عملية فك التطبيع، إلا أن هذه السلبيات تتلاشى إذا ما تمَّت العملية بشكل دقيق ونتج عنها زيادة كبيرة في سرعة معالجة البيانات.

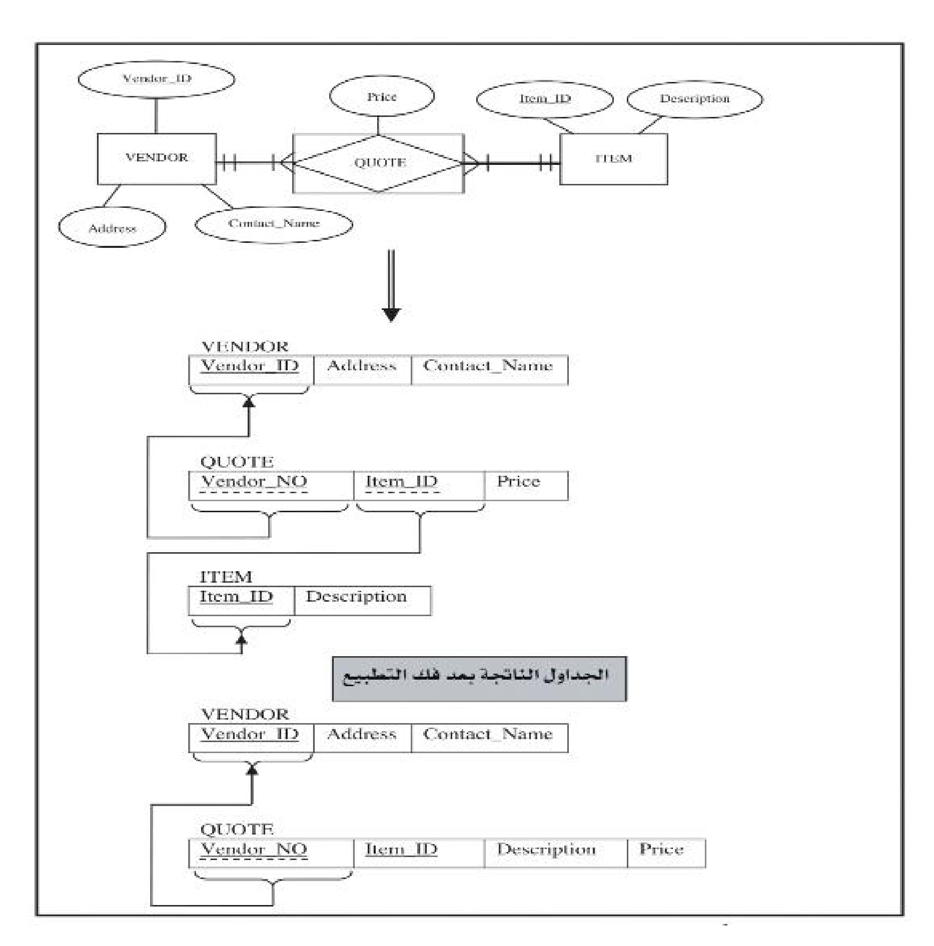
ويُوجَد هناك عددٌ من الحالات المعروفة التي قد تتطلب إجراء عمليات فكِّ التطبيع، وفيما يلي ثلاث من هذه الحالات (Rogers, 1989):

1- جدولان تربط بينهما علاقة «واحد - واحد»: يُفضَّل فكُّ التطبيع في هذه الحالة حتى لو كان أحد الجدولين ذا علاقةٍ اختيارية؛ وذلك عندما يكون هناك ارتباط بين سجلات الجدولين في غالبية الأحيان. ويتمُّ في هذه الحالة إنشاءُ سجل واحد ضمن جدول واحد عوضاً عن سجلين في جدولين مختلفين. وبهذه الطريقة يتمُّ الاستغناء عن عمليات الربط التي يجب إجراؤها للحصول على البيانات المخزَّنة في الجدولين للحصول على بيانات السجلات التي تربط بينهما العلاقة. ويوضِّح الشكل رقم (6-8) مثالاً لمثل هذه الحالة؛ إذ يُوجَد لكلِّ موظف موقف سيارات واحد على الأكثر، وأن كل موقف سيارات مخصص لموظف واحد على الأكثر.



شكل رقم (6–8): فكُّ التطبيع بين جدولين تربط بينهما علاقة «واحد – واحد».

2- علاقة «متعدد - متعدد» (أي: علاقة مشاركة) بين جدولين ولا يُوجَد للعلاقة مفتاح خاص بها: قد يكون من المفيد دَمْجُ حقول أحد الجدولين اللذين تربط بينهما العلاقة؛ ليكونا من ضِمن الجدول الذي يمثل العلاقة. وبهذه الطريقة يمكِن إجراء عملية ربط واحدة بين الجدولين الناتجين عوضاً عن إجراء عمليتَيْ ربط بين ثلاثة جداول. وتتجلى أهمية هذه الطريقة عندما تكون غالبية العمليات المنفذة على الجداول الثلاثة تتطلب عمليات ربط بينها. ويوضِّح الشكل رقم (6-9) مثالاً لهذه الحالة.

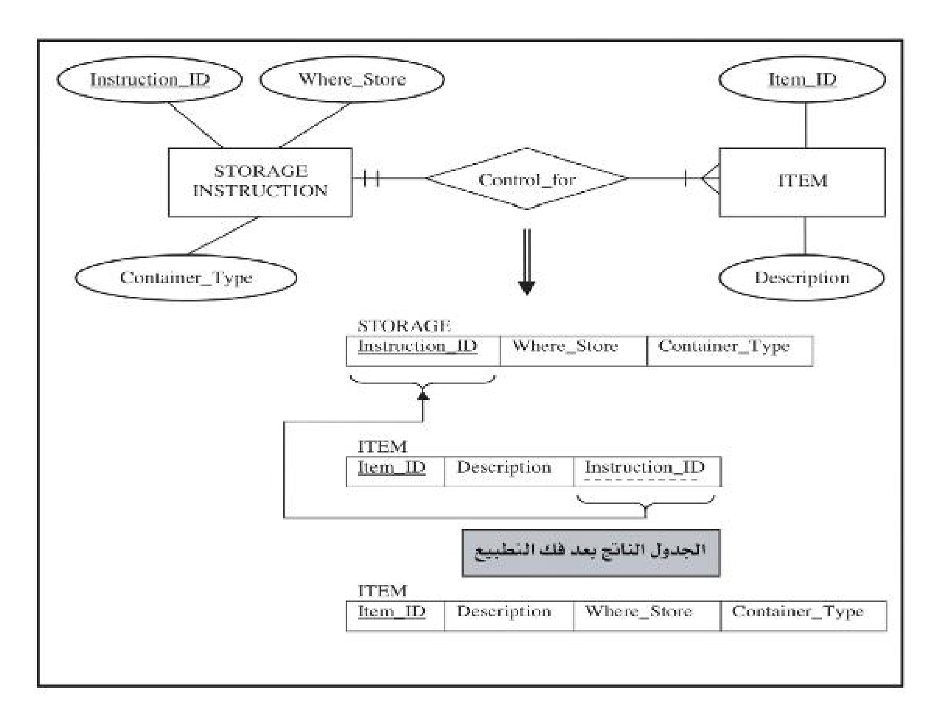


شكل رقم (6–9): فكُّ التطبيع عند وجود علاقة «متعدد – متعدد»

(أيْ: كينونة مشاركة) بين جدولين ولا يوجد للعلاقة مفتاح خاص بها.

3- بيانات مرجعية: توجد البيانات المرجعية عندما تكون هناك كينونة في الجانب ذي التعددية الأحادية من علاقة ذات تعددية «واحد - متعدد»، وهذه الكينونة لا ترتبط بأيَّة علاقات أخرى مع الكينونات الأخرى المكوّنة لقاعدة البيانات. في هذه الحالة يُفضَّل دَمْج الجدولين الممثلين للكينونتين ضِمن جدولٍ واحد؛ وخاصةً عندما يكون عدد الحالات في الجانب المتعدد المرتبطة بالجانب الأحادي قليلة نسبياً.

يوضِّح الشكل رقم (10-6) مثالاً للحالة المذكورة أعلاه؛ إذ إن كل «عنصر» (ITEM) له «طريقة في التخزين» (STORAGE_INSTRUCTION) تتمثل في «مكان تخزين العنصر» (Where_Store) و«نوع الإناء» (Container_Type). وعندما يكون عدد حالات العناصر قليلاً نسبياً، وتكون «طريقة التخزين» (STORAGE_INSTRUCTION) مرتبطة بكينونة «العنصر» (ITEM) فقط؛ فإنه يُفضَّل دمج جدول العناصر مع جدول طريقة التخزين ضِمن جدول واحد وهو جدول «العنصر» (ITEM).



شكل رقم (10–6): فكُّ التطبيع عند وجود بيانات مرجعية.

إن المواقع السابقة التي قد تستدعي عمليات فك التطبيع؛ تهدف إلى الاستغناء عن عمليات ربط بين جداول قاعدة البيانات التي تستنزف الوقت الكثير لتنفيذها؛ وخاصةً عند احتواء الجداول على أعداد كبيرة من السجلات. ويتمُّ ذلك من خلال دَمْج الجداول بعضها مع بعض. وعلى النقيض من ذلك؛ فإنه يمكن فكُّ التطبيع من خلال تقسيم جدولٍ ما إلى أكثر من جدول عوضاً عن دمجه مع جدول آخر كما في الحالات السابقة. وهناك ثلاثة أنواع من تقسيم الجداول، وهي: التقسيم الأفقي، والتقسيم الرأسي، والتقسيم الذي يمزج بين الاثنين (أفقياً ورأسياً معاً). ويهدف التقسيم الأفقي لجدول ما إلى تجميع السجلات التي تشترك في قيمة خاصية (أو حقل) معين ضِمن جدولٍ واحد؛ بحيث تتمُّ غالبية الاستفسارات التي ترد للنظام وفقاً لقيمة هذه الخاصية. فعلى سبيل المثال: قد يتمُّ تقسيم جدول الموظفين العاملين في منظمة ما إلى أكثر من جدولٍ وفقاً للإدارات التي يعمل فيها الموظفون. في هذه الحالة يكون من الأسرع الاستجابة للتعليمات التي

ترد للنظام، والتي تتعامل مع سجلات الموظفين العاملين في كلِّ إدارة على حِدَة. كذلك هو الحال بالنسبة لجدول العملاء المتعاملين مع شركةٍ ما؛ إذ يُمكِن تقسيم ملف العملاء حسب المناطق التي يقطنها هؤلاء العملاء. وفي مثل هذه الحال؛ يكون من الأسرع الاستجابة للاستفسارات التي تتعامل مع العملاء وفق المناطق التي يقطنونها.

أمَّا التقسيمُ الرأسي؛ فيقوم بتوزيع الحقول التي يحتويها جدول ما على أكثر من جدول مع تكرار المفتاح الرئيسي للجدول الأساسي في جميع الجداول الناتجة من عملية التقسيم. ومن أمثلة التقسيم الرأسي تقسيم جدول الموظفين إلى جدولين: جدول يحتوي على المفتاح الرئيسي لجدول الموظفين؛ إضافةً إلى الحقول التي تحتوي على البيانات المتعلقة بالجوانب الإدارية للموظفين، في حين أن الجدول الثاني يحتوي على المفتاح الرئيسي لجدول الموظفين، بالإضافة إلى الحقول التي تحتوي على البيانات المتعلقة بالجوانب المالية للموظفين. وبهذه الطريقة يمكن الاستجابة للتعليمات المتعلقة بكل نوع من البيانات بشكلٍ أسرع من الرد عليها عند تخزين جميع البيانات المتعلقة بالموظفين ضِمن نفس الجدول.

ومن المَيْزات التي يتحلى بها كلا التقسيمين أعلاه؛ إعطاء صلاحيات للمستفيدين على الجداول الناتجة بعد عملية التقسيم؛ وذلك عوضاً عن إعطائهم صلاحيات على الجداول بشكلها الكامل (قبل عملية التقسيم)؛ مما يسهم في دقة إعطاء الصلاحيات، ومن ثم تحسين درجة أمن وسرية المعلومات.

أما النوع الثالث للتقسيم؛ فيتضمن التقسيم الأفقي والتقسيم الرأسي. فعلى سبيل المثال: من الممكن تقسيم جدول الموظفين أفقياً حسب الإدارات التي يعمل فيها الموظفون ورأسياً حسب بياناتهم الإدارية والمالية. وتتجَلَّى فائدة هذا النوع من التقسيم بشكلٍ خاص في قواعد البيانات الموزَّعة التي سنتطرق إليها في الفصل التاسع.

3–1–2–6 تنظيم الملفات (File Organization):

يُستخدَم تنظيم الملفات لترتيب السجلات التي ستُخزَّن في ملفات الذاكرة الثانوية. ويتم اختيار أحد تنظيمات الملفات في نظم إدارة قواعد البيانات لملف ما آخذين بعين الاعتبار سبعة عوامل مهمة، وهي (Hoffer et al, 2018):

- السرعة في استرجاع البيانات.
- السرعة في الحصول على نتائج معالجة التعليمات.
- الفعالية في استخدام الذاكرة.

- الحماية من الأعطال، أو فقد البيانات.

- التقليل من الحاجة إلى إعادة تنظيم السجلات في الملف.

- التمكُّن من التحكم في التوسع في حجم البيانات.

- تأمين البيانات من الاستخدامات غير المسموح (أو المصرَّح) بها.

وعادةً ما تتعارض العوامل السابقة بعضها مع بعض عند اختيار تنظيم معين للملفات ويجب اختيار التنظيم الذي يوفر التوازن المناسب بين المعايير السابقة؛ للاستفادة القصوى من المصادر المتاحة في النظام. وفيما يلي شرحٌ (مقتضب) لأهمِّ تنظيمين للملفات، وهما: الملفات المتسلسلة، والملفات المفهرسة.

الملفات المتسلسلة: يتمُّ في الملفات التي تنظم بشكلٍ متسلسلٍ (Sequential Files) تخزين السجلات بشكلٍ متسلسل، الواحد تلو الآخر، حسب قيمة مفتاح رئيسي (Primary Key). وللوصول إلى سجلٍ ما يتمُّ المرور على السجلات المخزَّنة في الملف من البداية، وبشكلٍ متسلسل، حتى الوصول إلى السجل ذي قيمة المفتاح المطلوب. ومن الأمثلة الشهيرة للملفات المتسلسلة دليل الهاتف الذي يتمُّ فيه ترتيب أسماء المشتركين في خدمة الهاتف أبجدياً (وبشكلٍ تصاعدي) حسب أسماء عائلاتهم. وعندما يتمُّ البحث عن رقم هاتف أحد المشتركين؛ فإنه لا بد أن يتم المرور على جميع السجلات التي تسبق اسم المشترك قيد البحث عن رقم هاتفه حتى الوصول إلى الاسم المطلوب. وتجدر الإشارة إلى أن هذا النوع من الملفات لا يتم استخدامه من قِبل نظم قواعد البيانات؛ وذلك لعدم مرونته في الوصول إلى السجلات بشكلٍ فعال. فعلى سبيل المثال: تحتاج نظم إدارة قواعد البيانات إلى فحص (n/2)، في المتوسط، من سجلات جدولٍ ما للوصول إلى السجل ذي المفتاح (k) عندما يكون عددُ السجلات المخزَّنة في الملف هي لجدول عدد سجلاته (n)؛ وذلك لأن السجلات التي يجري البحث عنها قد تكون في بدايات الملف أحياناً، وقد تكون في نهايات الملف أحياناً أخرى بحسب قِيم مفاتيحها. ولكون المتوسط العام للوصول إلى السجلات المطلوبة باستخدام الملفات المتسلسلة يُعَدُّ عالياً نسبياً مقارنةً بأنواع الملفات الأخرى؛ فإن هذا النوع من الملفات قد يتمُّ استخدامه لأغراض النسخ الاحتياطي؛ ولكنه لا يُستخدَم باعتباره طريقة لتخزين البيانات في قواعد البيانات.

الملفات المفهرسة: يتمُّ تنظيم السجلات في الملفات المفهرسة؛ إما بشكلٍ متسلسلٍ منظمٍ حسب قيمة مفتاح أوَّلي ما، أو دون تسلسل معين للسجلات في الملف. ويتمُّ بناء فهرس للملف يسمح بالوصول إلى السجلات المطلوبة في الملف. والفهرس (في أبسط صوره) هو جدول يحتوي على عمودين: أحدهما يحتوي على قِيم مفاتيح السجلات في الملف، في حين يحتوي العمود الثاني على مؤشرات (Pointers) تبيِّن مواقع (أو عناوين) السجلات في الملف. ويمكن تشبيه الملفات المفهرسة بفهارس المطبوعات العلمية

في المكتبات؛ إذ إن كلَّ فهرس من فهارس المكتبات يحتوي على مِفتاح، مثل: «اسم المؤلف»، أو «دار النشر»، أو «الموضوع»، ويحتوي على قيمة توضِّح موقع المطبوعة ضمن ثنايا المكتبة. وبذلك؛ فإن كلَّ سجلٍ في الفهرس يحتوي على قيمة لمفتاح ومؤشر يدلُّ على موقع السِّجل ذي نفس قيمة المفتاح في الملف. وعندما يكون المفتاح مفتاحاً أوَّلياً (أو رئيسياً)؛ فإن لكل سجل قيمة مفتاح تختلف عن بقية السِّجلات في الملف، ولا يمكن أن تتكرر. أما إذا كان من الممكن أن تتكرَّر قيمة المفتاح؛ فإن المفتاح يُسمَّى ثانوياً (SeCondary Key). فعلى سبيل المثال: يُعَدُّ مفتاح الفهرس المبني على أرقام الطلبة مفتاحاً أولياً؛ وذلك لعدم إمكانية تكرُّر أرقام الطلبة، في حين يُعَدُّ مفتاح الفهرس المبني على أسماء عائلات الطلبة مفتاحاً ثانوياً؛ لأنه من الممكن أن تتكرَّر أسماء عائلات الطلبة.

وتكمُنُ أهمية الفهارس في نظم إدارة قواعد البيانات، بشكلٍ عام، في كونها تقلِّص من حجم البيانات الواجب البحث فيها؛ حتى يتمَّ الوصول إلى السجلات المطلوبة؛ إذ إن حجم أيِّ فهرس يكون عادةً أقل بكثير من حجم الملف الذي بني عليه الفهرس؛ وذلك لكون سجلات الملف عادةً ما تكون أطول بكثير من سجلات الفهرس. ويعني هذا أنه، وفي غالبية الأحيان، يمكن أن يُوضَع الفهرس بالكامل في الذاكرة الرئيسية للحاسب الآلي، والبحث عن مواقع السجلات من خلال البحث في الفهرس. أمَّا في حالة عدم وجود فهرس؛ فإنه يجب الرجوع للذاكرة الثانوية للجهاز، ولمرات عديدة، لنقل أجزاء من الملف والبحث فيها؛ حتى يتمَّ الوصول إلى السِّجلات قيد البحث. والسبب وراء ذلك يعود إلى كِبر حجم الملفات التي تحتوي على سجلات البيانات التي يتعذر معها وَضْع مثل هذه الملفات في الذاكرة الرئيسية للحاسب الآلي دفعةً واحدة. ولأن الذاكرة الثانوية تكون عادةً مبنيةً على تقنيات تدخل فيها الحركة الميكانيكية، مثل: الأقراص الصلبة التي تُعَدُّ الأكثر شيوعاً في تخزين البيانات؛ فإن عملية الرجوع إلى الذاكرة الثانوية بشكلٍ متكررٍ تُعَدُّ مكلفةً جداً في عمليات البحث وتؤدي إلى الإطالة في الوقت اللازم؛ للوصول إلى السجلات المطلوبة؛ مما يؤثر في فاعلية نظام إدارة قاعدة البيانات بشكل عام.

وكما هو في فهارس المكتبات، على سبيل المثال، يمكن بناء أكثر من فهرس للملف نفسه. كما تتوافر أنواع عديدة من الفهارس التي بالإمكان بناؤها على ملفات البيانات؛ ولكننا لن ندخل في تفاصيل الأنواع المختلفة للفهارس هنا.

6–2–1–4 إنشاء واستخدام الفهارس:

تتطلب غالبية التعليمات التي تنفذها نظم إدارة قواعد البيانات الوصول إلى سجل أو أكثر يتحقق فيها شرط معين. ومن أمثلة هذه التعليمات معرفة أعضاء هيئة التدريس الذين يعملون في قسم الحاسب الآلي، أو معرفة الطلبة الذين يدرسون في تخصص معين، أو معرفة الطلبة ذوي المعدلات التراكمية التي تكافئ

«جيد جداً» أو أكثر. وللحصول على نتيجة مثل هذه التعليمات؛ يجب المرور على جميع سجلات الملف الذي يحتوي على السجلات المطلوبة، الواحد تلو الآخر، ومقارنة الحقول التي ترتبط بشرط البحث مع القيمة المفروضة على شرط البحث. وتُعَدُّ عملية المرور على جميع سجلات الملف؛ بهدف الوصول إلى النتيجة المطلوبة عملية مكلفة جداً، وخاصةً عندما يكون عدد السجلات في الملف قيد البحث كبيراً جداً. لذلك يتمُّ استخدامُ الفهارس التي تمَّ شرحها (بشكلٍ مقتضب) أعلاه للتسريع في تنفيذ مثل هذه التعليمات التي تمثل غالبية التعليمات التي تنفَّذ على قاعدة البيانات.

1-4-1-2-6 إنشاء الفهارس ذات المفاتيح الفريدة (**Unique Key Index**):

يُمكِن إنشاء الفهارس ذات المفاتيح الفريدة على أيِّ حقل في جدول؛ بحيث إن قيمة الحقل الذي سينشأ عليه الفهرس لا يمكن أن تتكرر في سجلات الجدول. فعلى سبيل المثال: يمكن إنشاء فهرس فريد على حقل رمز المادة الدراسية (Course_ID) في جدول المواد الدراسية (COURSE) في قاعدة بيانات الجامعة الأهلية الذي يمثل المفتاح الرئيسي للجدول، كما يلي:

```
CREATE UNIQUE INDEX COURSE_ID_IDX ON COURSE(Course_ID);
```

وتمثل العبارة (COURSE_ID_IDX) اسم الفهرس الفريد الذي سيتمُّ إنشاؤه لحفظ مكوِّنات الفهرس، في حين يمثل ما بعد العبارة (ON) اسم الجدول الذي سيتمُّ بناء الفهرس عليه، والحقل الذي سيكون مفتاح الفهرس. وعندما تُنفَّذ تعليمة الإنشاء السابقة؛ ستتمُّ فهرسة جميع سجلات جدول المواد الدراسية ضِمن الفهرس. أمَّا إذا وُجِدَ أكثرُ من سجلٍ لها قيمة رقم المادة الدراسية نفسها (أيْ: القيمة نفسها في الحقل الذي يمثل مفتاح الفهرس) فستفشل عملية إنشاء الفهرس. وعند إنشاء الفهرس، في حال عدم وجود سجلات تتكرَّر فيها قيمة مفتاح الفهرس؛ فإنه سيتمُّ أيضاً رفض أيِّ عملية إضافة للجدول ينتج عنها سجلات تتكرر فيها قيمة مفتاح الفهرس.

وعندما يُراد إنشاء فهرس فريد ذي مفتاح مركب من أكثر من حقل؛ فإنه يمكن ذلك من خلال إدراج جميع الحقول المكونة للمفتاح بعد عبارة (ON) التي تتضمَّن اسم الجدول. فعلى سبيل المثال: يمكن إنشاء فهرس فريد على حقل رمز المادة الدراسية (Course_ID) ورمز المادة الدراسية المتطلبة (Prerequisite_ID) في جدول المواد الدراسية المتطلبة (PREREQUISITE) اللذين يمثلان المفتاح الرئيسي لجدول، كما يلي:

```
CREATE UNIQUE INDEX COURSE_PREREQUISITE_IDX
ON PREREQUISITE(Course_ID, Prerequisite_ID);
```

وتقوم غالبية نظم إدارة قواعد البيانات العلاقية بإنشاء فهرس فريد، وبشكلٍ تلقائي، لكلِّ جدول يتم إنشاؤه؛ بحيث يكون مفتاح الفهرس هو المفتاح الرئيسي للجدول. ويعني هذا عدم الحاجة إلى تنفيذ تعليمة إنشاء فهرس إذا كان الحقل الذي سينشأ عليه الفهرس هو المفتاح الرئيسي للجدول، مثل الفهارس التي تمَّ إنشاؤها أعلاه.

6-2-1-4-2 إنشاء الفهارس ذات المفاتيح الثانوية (أو غير الفريدة) (SeCondary Key Index):

تُستخدَم الفهارس ذات المفاتيح الثانوية عندما يُراد التعامل مع حقول غير الحقول الممثلة للمفاتيح الرئيسية للجداول. فعلى سبيل المثال: قد يتعامل المستفيدون مع جدول الطلبة (STUDENT) في قاعدة بيانات الجامعة الأهلية من خلال تعليمات تحاول التعرف على أسماء الطلبة في تخصصات معينة، أو قد يتمُّ التعامل مع جدول المتطلبات الدراسية من خلال تعليمات تحاول التعرُّف على المتطلبات الدراسية للمواد الدراسية المختلفة. وللتسريع من عملية تنفيذ مثل هذه التعليمات؛ فإنه من الممكن إنشاء فهرس على كل حقل من الحقول المضمَّنة في شرط الاسترجاع. فعلى سبيل المثال: يمكن إنشاء فهرس (غير فريد) على حقل تخصصات الطلبة باستخدام التعليمة التالية:

```
CREATE INDEX MAJOR_IDX ON STUDENT(Major);
```

6-2-1-4-3 استخدامات الفهارس:

يجب في مرحلة التصميم المادي لقاعدة البيانات العنايةُ في اختيار الحقول التي سيتم استخدامها في إنشاء الفهارس؛ وذلك لوجود عملية موازنة (trade-off) بين تحسين أداء نظام إدارة قاعدة البيانات في تنفيذ تعليمات الاسترجاع وبين خفض أداء النظام في تنفيذ عمليات الإضافة والحذف والتحديث؛ وذلك لكون العمليات الثلاث الأخيرة قد تتطلب تعديل محتويات الفهارس المبنية على الجداول التي تتعامل معها هذه العمليات. ومثل عمليات التعديل هذه في محتويات الفهارس تضيف المزيد من التكلفة عند تنفيذ عمليات التعديل على محتويات قاعدة البيانات. لذلك؛ فإنه يتمُّ عادةً الاستخدام المكثف للفهارس عندما تتسم الغالبية العظمى من العمليات المنفذة على قاعدة البيانات في كونها عمليات استرجاع. أمَّا إذا كانت الغالبية

العظمى من العمليات التي تنفذ على قاعدة البيانات تتسم بكونها عمليات تعديل؛ فإنه يجب أخذ الحيطة والحكمة عند إنشاء الفهارس. وفيما يلي بعض القواعد العامة التي تساعد على اختيار وإنشاء الفهارس:

- تُعَدُّ الفهارس مفيدةً للجداول الكبيرة (ذات السجلات الكثيرة في عددها).

- تُعَدُّ الفهارس مفيدةً للحقول التي تظهر في عبارة شرط الاسترجاع (Where) من تعليمة الاختيار (SeleCt)، أو في عبارات الربط.

- تُعَدُّ الفهارس مفيدةً للحقول التي تظهر في عبارة الترتيب (Order By)، وعبارة التصنيف (Group BY)، مع ضرورة التأكد من أن نظام إدارة قاعدة البيانات سيستخدم فعلياً الفهارس المنشأة لهذا الغرض عند تنفيذه لتعليمات الاسترجاع التي تحتوي على عبارات الترتيب وعبارات التصنيف؛ إذ إن بعض نظم إدارة قواعد البيانات قد لا تستخدم الفهارس في مثل هذه التعليمات.

- تُعَدُّ الفهارس مفيدةً للحقول التي يزيد عدد القيم التي من الممكن أن تُخزَّن فيها عن ثلاثين (30) قيمة، وأن عدد السجلات المسترجعة لا تزيد على عشرين في المائة (20%) من عدد السجلات الكلية المخزنة في الملف.

- يجب التأكدُ من الحَدِّ الأقصى لعدد الفهارس التي من الممكن أن تنشأ لكلِّ جدول؛ إذ إن غالبية نظم إدارة قواعد البيانات لا تسمح بأن يزيد هذا العدد على ستة عشر (16) فهرساً. كما يجب معرفة الحَدِّ الأقصى لطول مفتاح الفهرس عند استخدام المفاتيح المركبة للفهارس. فعلى سبيل المثال: قد لا يسمح نظام إدارة قاعدة البيانات أن يزيد طول مفتاح الفهرس على مائة (100) بايت.

- يجب الحذر عند فهرسة حقول تسمح بالقيم الغائبة (NULL)؛ إذ إن غياب القيم لن يُمكِّن من تضمينها كمفاتيح للفهرس؛ وذلك في غالبية نظم إدارة قواعد البيانات؛ الأمر الذي يستوجب المرور على سجلات الملف كافة على الرغم من وجود فهرس للجدول.

وتُعَدُّ عملية اختيار الفهارس واحدةً من أهمِّ العمليات في مرحلة التصميم المادي، إلا أنها ليست العملية الوحيدة؛ فمن العمليات الأخرى التي من شأنها تحسين أداء نظام إدارة قاعدة البيانات عملية تقليص المساحة التخزينية المهدرة في الملفات، ومن ثم تقليص حجم الملفات الذي يترتب عليه تقليصُ عدد المرات التي يجب فيها الرجوعُ إلى الذاكرة الثانوية لنقل كلِّ ملف إلى الذاكرة الرئيسية عند الحاجة إلى ذلك؛ وعملية تقليص تكلفة تنفيذ الاستفسارات (Query Optimization) التي يُستخدَم فيها خوارزميات مُخصَّصة لهذا الغرض؛ إلا أننا لن نتطرق إلى هذين الموضوعين في هذا الكتاب.

الفصل السابع

لغة الاستفسار البنائية – الجزء الأول

تُعَدُّ لغة الاستفسار البنائية (StruCtured Query Language (SQL)) واحدةً من أكثر لغات قواعد البيانات العلاقية انتشاراً. وقد تمَّ تبنِّي هذه اللغة من قِبل معهد المقاييس الوطني الأمريكي (AmeriCan National Standards Institute (ANSI)) ومنظمة المقاييس الدولية (International Standards Organization (ISO)). وقد تم نشرُ أول مقياس لهذه اللغة من قِبل معهد المقاييس الوطني الأمريكي عام 1986م (Cannan and Otten, 1993)). وتمَّ تحديث مقياس لغة الاستفسار البنائية عام 1989م وعام 1992م وعام 1999م، 2003م، 2006م، 2008م، 2011م، 2016م، على التوالي. ويلخِّص الجدول رقم (7-1) التسلسل الزمني لهذا المقياس وبعض المزايا أو التحسينات التي أضافتها كلُّ نسخة منه على النسخة السابقة لها.

جدول رقم (7–1): التسلسل الزمني لتطوير مقاييس SQL.

السنة	اسم المقياس	الملاحظات
1986	SQL-82	أول نسخة رسمية يتمُّ اعتمادها من قِبل معهد المقاييس الوطني الأمريكي (AmeriCan National Standards Institute (ANSI))
1989	SQL-89	مراجعة طفيفة للمقياس السابق، وإضافة قيود التكامل (Integrity Constraints).
1992	SQL-92	مراجعة كبيرة للمقياس تضمَّنت إضافة أنواع بيانات جديدة،

وعمليات عددية للتعامل مع السلاسل الحرفية، على سبيل المثال لا الحصر.		
إضافة الاستعلامات المتداخلة (ReCursive Quires)، والزنادات (Triggers)، وبعض من خصائص النموذج الشيئي، ودعم SQL المضمَّن (Embedded SQL) مع لغة البرمجة جافا.	SQL: 1999	1999
أصبح المقياس مكوناً من تسعة أجزاء؛ بحيث تمَّ إضافة خصائص تتعلق بالإكس أم إل (XML)، ومولد المسلسلات (SequenCe Generator)، وإضافة تعليمة الدَّمج (Merge Operation)، وإضافة امتدادات لتعليمة إنشاء جدول.	SQL: 2003	2003
تمَّ وَضْع إضافات للجزء المرقم 14 من المقياس السابق والمتعلق باستخدام SQL مع XML.	SQL: 2006	2006
تمَّ إجراء بعض التحسينات على تعليمة الدمج (Merger Statement)، و(TrunCate Statement)، والزنادات، وتعليمة الربط (Join Statement).	SQL: 2008	2008
ركَّزت هذه النسخة من المقياس على إضافة تعليمات تتعلق بالبيانات الوقتية (Temporal Data).	SQL: 2011	2011
إضافة دوال تتعلق بالوثائق (DoCuments)، والتعرُّف على الأنماط (Pattern ReCognition).	SQL: 2016	2016

ويتكوَّن المقياس حالياً من تسعة أجزاء، كما هو مُوضَّح في الجدول رقم (7-2).

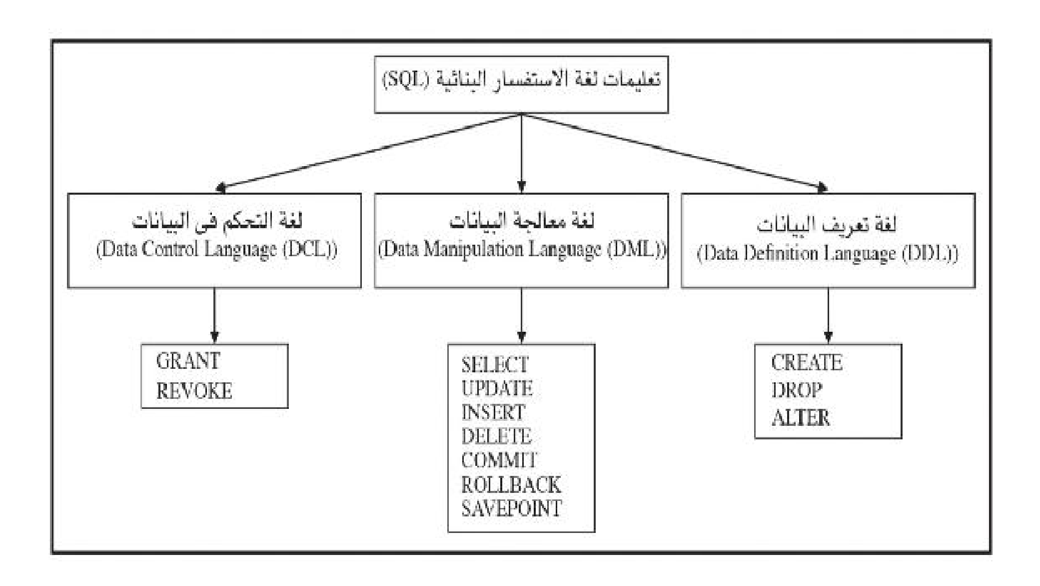
جدول رقم (7-2): الأجزاء المكوِّنة لمقياس لغة الاستفسار البنائية الحالي (2016م).

الرقم المرجعي	**عنوان الوثيقة**
ISO/IEC 9075-1	Information teChnology -- Database languages -- SQL -- Part 1: Framework (SQL/Framework)

ISO/IEC 9075-2	Information teChnology -- Database languages -- SQL -- Part 2: Foundation (SQL/Foundation)
ISO/IEC 9075-3	Information teChnology -- Database languages -- SQL -- Part 3: Call-Level InterfaCe (SQL/CLI)
ISO/IEC 9075-4	Information teChnology -- Database languages -- SQL -- Part 4: Persistent stored modules (SQL/PSM)
ISO/IEC 9075-9	Information teChnology -- Database languages -- SQL -- Part 9: Management of External Data (SQL/MED)
ISO/IEC 9075-10	Information teChnology -- Database languages -- SQL -- Part 10: ObjeCt language bindings (SQL/OLB)
ISO/IEC 9075-11	Information teChnology -- Database languages -- SQL -- Part 11: Information and definition sChemas (SQL/SChemata)
ISO/IEC 9075-13	Information teChnology -- Database languages -- SQL -- Part 13: SQL Routines and types using the Java™ programming language (SQL/JRT)
ISO/IEC 9075-14	Information teChnology -- Database languages -- SQL -- Part 14: XML-Related SpeCifiCations (SQL/XML)

وعلى الرغم من إطلاق كلمة «الاستفسار» على لغة التعامل مع قواعد البيانات العلاقية؛ فإنَّ هذه الكلمة مجازيةٌ؛ بمعنى أن مكوِّنات لغة الاستفسار البنائية لا تنحصر في التعليمات التي تقوم بعمليات الاستفسار؛ بل تتعَدَّى ذلك لتحتوي على ثلاث فئات من أنواع التعليمات. تُعْنَى الفئة الأولى بعمليات إنشاء مكوِّنات (أو هياكل) قاعدة البيانات، وإزالتها، والتعديل عليها. أما الفئة الثانية من أنواع التعليمات؛ فتُعْنَى بالعمليات التي تقوم بالتعامل الفعلي مع البيانات المخزَّنة في قاعدة البيانات، مثل: عمليات الإضافة إليها، والحذف منها، والتحديث عليها. الفئة الثالثة من أنواع التعليمات تُعْنَى بعمليات التحكُّم في تداول البيانات من قِبل المستفيدين؛ من خلال تزويدهم

بالصلاحيات المناسبة لتداولها أو سحب الصلاحيات منهم. وعلى الرغم من أن كلَّ الفئات الثلاث من التعليمات تتبع للغةٍ واحدة وهي لغة الاستفسار البنائية؛ فإن هذه الفئات تُسمَّى مجازاً لغات فرعية (Sub-languages) للغة الاستفسار البنائية. ويُوضِّح الشكل رقم (7-1) اللغات الفرعية (أو الفئات) الثلاث المكوِّنة للغة الاستفسار البنائية.



شكل رقم (7–1): اللغات الفرعية (أو الفئات الثلاث من التعليمات) المكوِّنة للغة الاستفسار البنائية.

ونظراً لأهمية لغة الاستفسار البنائية في قواعد البيانات العلاقية؛ فإن هذا يستلزمُ شرحَ مكوِّناتها الأساسية شرحاً تطبيقياً مستفيضاً؛ حتى يتسنَّى فهم طريقة عمل تعليماتها. لذا فقد تمَّ تخصيص هذا الفصل والفصل التالي لشرح المكوِّنات الأساسية للغة الاستفسار البنائية. ويحتوي هذا الفصل على شرح لتعليمات لغة تعريف (هياكل) البيانات، وعلى شرح للعبارات الأساسية في تعليمة الاختيار (أو الاسترجاع) التي تُعَدُّ من أهمِّ تعليمات لغة معالجة البيانات. أمَّا الفصل التالي؛ فيستكمل شرح مكونات لغة الاستفسار البنائية؛ بحيث خُصِّص الجزء الأول منه لاستكمال شرح تعليمة الاختيار، عندما تقوم التعليمة بالتعامل مع أكثر من جدول في آنٍ واحد، ولشرح بقية تعليمات لغة معالجة البيانات. أمَّا الجزء الثاني من الفصل الثامن؛ فقد خُصِّص لشرح تعليمات لغة التحكُّم في البيانات.

وسنستخدمُ في هذا الفصل والفصل الثامن نظام إدارة قاعدة بيانات أوراكل في تنفيذ واختبار تعليمات لغة الاستفسار البنائية القياسية؛ وذلك لكون هذه البيئة واحدةً من الأوسع انتشاراً بين

المتخصِّصين في تطوير التطبيقات المبنية على نظم قواعد البيانات، هذا بالإضافة إلى تشابهها مع ما توفره نظم إدارة قواعد البيانات المعروفة الأخرى على المستوى التجاري، مثل: «دي بي 2» (DB2)، و«سايبيس» (SYBASE) من بيئات مشابهة لتنفيذ تعليمات لغة الاستفسار البنائية. أمَّا بالنسبة لمَنْ لا تتوفَّر لديهم بيئة أوراكل، وحتى يتمكَّن هؤلاء من تطبيق بعض مفاهيم وتعليمات لغة الاستفسار البنائية، والاستفادة القصوى من محتويات هذين الفصلين؛ فنستخدم قاعدة بيانات أكسس (ACCESS)، وبشكلٍ مقتضب؛ وذلك لتوافرها في غالبية الحاسبات الشخصية في وقتنا الراهن.

7–1 لغة تعريف البيانات (Data Definition Language (DDL)):

7–1–1 تعليمة الإنشاء (Create Statement):

يتكوَّن مقياسُ لغة الاستفسار البنائية التي تمَّ وَضْع مقاييسها عام 1992م (SQL-92) من ثلاثة مستويات، وهي: الأساسي (Entry)، والمتوسط (Intermediate)، والكامل (Full). ويحتوي المقياسُ في مستواه الأساسي على ثلاثة أنواع من تعليمات الإنشاء: إنشاء قاعدة بيانات، وإنشاء جدول، وإنشاء منظور. إضافةً إلى ذلك؛ فإن مقاييس لغة الاستفسار البنائية (SQL-92) تحتوي على خمسة أنواع أخرى من تعليمات الإنشاء في مستواها المتوسط، من ضِمنها تعليمة إنشاء القيود (Create Assertion)، وتعليمة إنشاء المجال (Create Domain). ويُلاحَظ أن تعليمة إنشاء فهرس (Create Index) قد تمَّ حذفها من المقاييس الحديثة على الرغم من تضمينها في المقاييس القديمة؛ وذلك لكون الفهارس تتعلق بعمليات تحسين أداء قاعدة البيانات ولا تُعَدُّ جزءاً من عمليات إنشاء قواعد البيانات العلاقية، أو تداول محتوياتها. وفيما يلي شرحٌ لتعليمات إنشاء قاعدة بيانات، وإنشاء جدول، وإنشاء مجال، وإنشاء منظور، وإنشاء فهرس، على التوالي.

7–1–1–1 إنشاء قاعدة بيانات (Create SChema):

لم تحتوِ المقاييس السابقة لمقياس (1992) على مبدأ تعريف هياكل لأكثر من قاعدة بيانات واحدة، وإنما كانت جميع تعاريف العلاقات والمكوِّنات الأخرى جزءاً لهيكل واحد لقاعدة البيانات. غير أن مقياس (1992) أوجد مبدأ هيكل قاعدة البيانات الذي يتمُّ من خلاله تجميع جميع العلاقات

المرتبطة فيما بينها والمكوِّنات الأخرى لقاعدة البيانات الواحدة ضِمن هيكل واحد. ويُعرَف هيكل قاعدة البيانات من خلال استخدام التعليمة التالية:

```
CREATE SCHEMA Database_Name AUTHORIZATION
Owner_UserID;
```

وتُمثِّل الكلماتُ المكتوبة بالأحرف الكبيرة وبالخط الداكن كلماتٍ محجوزةً يجب أن تتضمَّنها تعليمة إنشاء هيكل قاعدة البيانات. أمَّا الكلمة (Database_Name)؛ فتمثل اسم هيكل قاعدة البيانات قيد الإنشاء، والكلمة (Owner_UserID) تمثل رمز دخول المستخدِم الذي سيكون مالكاً لهيكل قاعدة البيانات، والذي من حقِّه التصرُّف بمحتوياتها وإعطاء الصلاحيات عليها. وعادةً يتمُّ استخدام تعليمة إنشاء هيكل قاعدة البيانات السابقة من قِبل إداريِّ قاعدة البيانات في المنشأة فقط. ولا شك أن مبدأ استخدام تعريف هياكل قواعد البيانات مفيدٌ جداً في الغالبية العظمى من المنظمات. فعلى سبيل المثال: يتمُّ عادةً استخدام هيكلين من قواعد البيانات في غالبية المنظمات التي تقوم بتطوير النظم التطبيقية التي تتعاملُ مع قاعدة البيانات. أحد هذين الهيكلين مخصصٌ لاختبار التطبيقات وقاعدة البيانات (Testing Environment)، والثاني مخصصٌ للنظم العاملة فعلياً في المنظمة (ProduCtion Environment).

ومثالٌ على البيئة التي تمَّت الإشارة إليها أعلاه، يمكِن تعريف قاعدة البيانات الاختبارية، كما يلي:

```
CREATE SCHEMA Company_Testing AUTHORIZATION
Saleh_Ahmed;
```

حيث إن اسم هيكل قاعدة البيانات الاختبارية هو (Company_Testing)، وأن الشخص صاحب الصلاحية الكاملة عليها هو «صالح أحمد» الذي لديه رمز المستخدم (Saleh_Ahmed). ويمكِن إنشاء هيكل قاعدة البيانات في البيئة التشغيلية (أو الإنتاجية) بالطريقة نفسها، وكما يلي:

```
CREATE SCHEMA Company AUTHORIZATION
```

```
Mohammed_Ali;
```

وبعد إنشاء هيكلي قاعدتي البيانات يُمكِن إنشاءُ هياكل العلاقات والمكوّنات الأخرى لكلٍّ منها، وبحيث يمكِن أن تتطابق مكوِّنات كلا الهيكلين بـما في ذلك مُسمَّيات تلك المكونات في كلا الهيكلين. ويعني هذا أن مقياس 1992 يُمكِّن من إنشاء أكثر من هيكل عوضاً عن هيكل واحد فقط يحتوي على مكونات كلا الهيكلين السابقين. ويزوِّد هذا المبدأ إداريِّ قواعد البيانات والمستفيدين منها بمرونة كبيرة. فعلى سبيل المثال: يمكن تحويل قاعدة البيانات الاختبارية إلى البيئة التشغيلية بمجرد تغيير مُسمَّى هيكل قاعدة البيانات وتزويدها ببيانات المنظمة الموجودة أصلاً في البيئة التشغيلية السابقة دون الحاجة إلى تغيير مُسمَّيات أو مكوِّنات أيٍّ من العلاقات والمكونات الأخرى في هيكل قاعدة البيانات الاختبارية. ومثل هذه العملية لا يمكِن أن تتمَّ بهذه البساطة دون مبدأ تعريف هياكل قواعد البيانات؛ إذ إن كلَّ مكونٍ لها في البيئة التشغيلية يجب أن يأخذ اسماً مختلفاً عن البيئة الاختبارية. ويتفاقم هذا الوضع ويتعقَّد في حالة وجود العديد من المكوِّنات التابعة منطقياً لهياكل مختلفة من قواعد البيانات؛ ولكنها مُعرَّفة ضِمن هيكل واحد.

7–1–1–2 إنشاء جدول (Create Table):

تُستخدَم تعليمة إنشاء جدول (Create Table) لتعريف علاقة جديدة ضِمن هيكل قاعدة البيانات. وتتضمَّن التعليمة اسم العلاقة وتعريفاً للحقول التي تحتويها وأيَّة قيود مفروضة عليها، كما يوضِّح الشكل العام التالي للتعليمة:

```
CREATE TABLE TableName
(Column–definition [,Column–definition] );
```

حيث إنَّ (Column-Definition) في الشكل العام للتعليمة يمثِّل تعريفاً للحقول المكوِّنة للعلاقة، وأن كلَّ حقل يجب أن يرتبط بمُسمَّى معين وبنوع واحد من أنواع البيانات التي يمكِن أن يحتوي عليها الحقل وأيَّة قيود مفروضة على الحقل كما هو مُوضَّح في الشكل التالي:

```
Constraint]ColumnName      data–type      [
```

كما يُمكِن إضافة المفتاح الرئيسي، والمفاتيح الخارجية التي تمثل قيود السلامة المرجعية ضِمن تعليمة الإنشاء، وبعد تعريف حقول العلاقة، أو تعريفها لاحقاً من خلال استخدام تعليمة «تغيير» (أو «تعديل») العلاقة (Alter Table).

ومن القواعد التي يجب مراعاتها عند تعريف أيَّة علاقة ما يلي:

- يجب ألا يتكرَّر اسم أيِّ حقل (Column Name) في الجدول نفسه.

- يجب أن يكون نوع بيانات أيِّ حقل (Data Type) من ضِمن الأنواع المعروفة لقاعدة البيانات (مثل: السلاسل الحرفية، الأرقام الصحيحة... إلخ) أو الأنواع التي يقوم مصممُ قاعدة البيانات بـتعريفها.

- توفر لغة الاستفسار البنائية عِدَّة أنواع من القيود (Constraints) التي يمكِن أن تُفرَض على حقول الجداول والتي يأتي من ضِمنها القيد على القيم الغائبة (Not Null) الذي يُقصَد منه أنه لا بد أن يتواجد في الحقل قيمةٌ في حالات الجدول كافةً، و(Unique) الذي يُقصَد منه أن قيمة الحقل يجب أن تكون فريدةً تُمايز بين كافة السجلات المخزَّنة في الجدول (أو العلاقة)، و(Primary Key) الذي يُقصَد منه أن الحقل هو المفتاح الرئيسي للجدول. وعند فرض قيد المفتاح الرئيسي على حقلٍ ما؛ فإن هذا يعني ضِمنياً أن الحقل ذو قِيَمٍ فريدة (Unique)، وأنه لا يمكِن أن تكون أيٌّ من قِيمهِ غائبةً (Null). بالإضافة إلى ذلك يُوجَد أنواعٌ أخرى من القيود التي سوف نتطرق إليها لاحقاً في هذا الفصل.

ومثال على تعليمة إنشاء جدول؛ لنفترض أننا نرغب في إنشاء جدول الأقسام العلمية في الجامعة الأهلية بمُسمَّى (Department_T) الذي يمكِن إنشاؤه حسب تعليمة الإنشاء التالية:

```
CREATE TABLE DEPARTMENT_T
    (DEPARTMENT_ID      CHAR(6)        NOT NULL,
     NAME               CHAR(30)       NOT NULL,
     CONSTRAINT DEPARTMENT_PK PRIMARY KEY (DEPARTMENT_ID));
```

يتكوَّن تعريف الجدول السابق من حقلين هما: رمز القسم (Department_ID)، واسم القسم (Name) وأن نوعية بيانات كلٍّ منهما عبارة عن سلسلة حرفية (Char)؛ غير أن طول السلسلة الحرفية للحقل الأول هي ستة أحرف، في حين أن طول سلسلة الثاني الحرفية ثلاثون حرفاً. كما يحتوي تعريف الجدول على ثلاثة قيود؛ اثنان منها يتعلقان بالحقل الأول والحقل الثاني؛ إذ إن كلا الحقلين يجب ألا تكون قِيمهما غائبةً، في حين أنَّ القيد الثالث يبيِّن المفتاح الرئيسي للجدول، وقد سُمِّي هذا القيد بمُسمَّى (Department_PK). ويعني هذا القيد أن المفتاح الرئيسي للجدول هو رمز القسم (Department_ID)، وأن مُسمَّاه هو (Department_PK). ويُعَدُّ تزويد القيود بمُسمَّيات من الأساليب الحميدة التي يتّبعها إداريّو قواعد البيانات عند إنشائهم لقواعد البيانات؛ إذ يُمكِّن هذا الأسلوب من التعرف على القيود المفروضة على قاعدة البيانات في وقتٍ لاحقٍ؛ من خلال استعراض كتالوج النظام. كما يُمكِّن أيضاً من إلغاء القيود وإعادة فرض قيود أخرى دون الحاجة إلى حذف الجدول (وكافة محتوياته)؛ ومن ثم إعادة تعريفه بقيود جديدة (أو مُعدَّلة). وفي حالة عدم الرغبة في إنشاء قيد المفتاح الرئيسي بمُسمَّى معين؛ فإنه يمكن كتابة القيد، كما يلي:

```
CREATE TABLE DEPARTMENT_T
        (DEPARTMENT_ID          CHAR(6)          NOT NULL,
         NAME                   CHAR(30)         NOT NULL,
         PRIMARY KEY (DEPARTMENT_ID));
```

أو يمكِن تعريف المفتاح الرئيسي مباشرةً كقيد على رمز القسم، كما يلي:

```
CREATE TABLE DEPARTMENT_T
        (DEPARTMENT_ID          CHAR(6)          PRIMARY KEY,
         NAME                   CHAR(30)         NOT NULL);
```

1-2-1-1-7 أنواع البيانات (**Data Types**):

توفر لغة الاستفسار البنائية بعض أنواع البيانات الأساسية، المضمَّنة في مقاييس اللغة؛ غير أنَّ غالبية نظم إدارة قواعد البيانات التجارية توفِّر أنواع بيانات إضافية لا تتضمَّنها مقاييس اللغة. ولذلك؛ فإننا نجد اختلافاً فيما توفره نظم قواعد البيانات التجارية من أنواع بيانات إضافية. ويأتي من ضِمن أنواع البيانات الأساسية التي تتضمَّنها لغة الاستفسار البنائية القياسية الأرقام، والسلاسل

الحرفية، والسلاسل المكوَّنة من الأرقام الثنائية (Bits)، والقِيم المنطقية الثنائية (Boolean)، والتاريخ، والوقت. وفيما يلي إيضاحٌ لأنواع البيانات الأساسية في لغة الاستفسار البنائية.

- الأرقام (NumeriC): تتكوَّن أنواع الأرقام من الأعداد الصحيحة (INTEGER)، والأرقام الصحيحة القصيرة (SMALLINT)، والأعداد الحقيقية (REAL) بمقاسات مختلفة من الدقة (PreCision)؛ إذ يُمكِن استخدام DeCimal(i, j) أو Number(i, j)؛ بحيث يدلُّ (i) على العدد الكلي للخانات العشرية (أو درجة الدقة)، في حين يدل (j) على عدد الخانات العشرية بعد الفاصلة العشرية. وتكون القيمة الافتراضية لعدد الخانات بعد الفاصلة العشرية في حالة عدم تعريفها صفراً.

- السلاسل الحرفية (CharaCter-String): يمكِن تعريف السلاسل الحرفية على أنها ثابتة أو متغيرة الطول. وتُعرَف السلاسل الحرفية ثابتة الطول بـ Char(n) أو CharaCter(n)؛ بحيث يرمز (n) للعدد الأكبر من الحروف التي من الممكن أن تحتويها السلسلة الحرفية. وفي حالة قل عددُ حروف السلسلة الحرفية للحقل عند إدخال البيانات فيه عن العدد الأكبر؛ فإنه يتمُّ إضافة حروف فاضية (Blank CharaCters)؛ لاستكمال السلسلة الحرفية حتى تصل للعدد (n). فعلى سبيل المثال: عند إدخال الاسم (Ahmed) في حقل الاسم لعلاقةٍ ما؛ فإنه سيخزَّن على أساس "Ahmed"؛ بحيث يتم إضافة خمسة حروف فاضية لاستكمال السلسلة الحرفية على افتراض تعريفها بأنها بطول عشرة أحرف. كما يجب استخدام علامتي التنصيص المفردة في أثناء عملية إدخال البيانات النصية. فعند إدخال الاسم (Ahmed) يجب أن يدخل "Ahmed".

أمَّا بالنسبة للسلاسل الحرفية المتغيرة الطول؛ فتُعرَف بـ VCHAR(n)، أو VARYING(n)، أو CHARACTER(n) VARYING، أو VARCHAR(n). ومرةً أخرى يرمز (n) للعدد الأكبر من الحروف التي من الممكن أن تحتويها السلسلة الحرفية المُدخَلة للحقل. ويَكمُن وجه الاختلاف بين السلسلة الحرفية الثابتة الطول والسلسلة الحرفية المتغيرة الطول؛ في كون الأخيرة تحتوي على العدد الفعلي للبيانات المُدخَلة دون إضافة حروف فاضية؛ مما ينتج عنه سجلات تابعة لنفس الجدول؛ ولكن بمقاسات مختلفة عوضاً عن كونها ثابتة المقاسات. كما تجدر الإشارة إلى أن السلاسل الحرفية حساسة لأحجام الحروف فالحرف "C" يُعَدُّ مختلفاً عن

الحرف "C" على سبيل المثال. ويعني هذا أن الاسم "Ahmed" مختلف عن الاسم "AHMAD" والاسم "AHmed".

- سلاسل الأرقام الثنائية (Bit-String): يُمكِن أن تكون سلاسل الأرقام الثنائية إما ثابتة الطول؛ بحيث تُعرَف على أساس BIT(n)، وإمَّا متغير الطول وتُعرَف على أساس BIT(n) VARYING. ويُقصَد بـ (n) في كلتا الحالتين الحدُّ الأقصى لطول سلسلة الأرقام الثنائية التي من الممكن إدخالها للحقل. والحالة الافتراضية عند عدم تحديد الحد الأقصى للسلسلة هو سلسلة رقمية ثنائية بطول واحد ((BIT(1)). وللتفريق بين السلاسل الحرفية والسلاسل الرقمية؛ فإنه يجب تحديد الحرف "B" قبل السلسلة الرقمية الثنائية، مثل: "10101100" B.

- القيم المنطقية الثنائية (Boolean): القيمة المنطقية الثنائية، وكما هو متعارف عليه في لغات البرمجة الأخرى؛ من الممكن إما أن تكون «صح» (True) أو أن تكون «خطأ» (False). غير أنه بسبب سماح لغة الاستفسار البنائية باستخدام القيمة الغائبة (Null)؛ فإن هذا يستدعي استخدام قيمة ثالثة وهي القيمة «غير المعلومة» (UNKNOWN) ضِمن ما يُعرَف بالمنطق الثلاثي القيم (True, False, Unknown)، الذي سنتطرق له لاحقاً (في الجزء 7-2-1-5-2-1).

- التاريخ (Date): نوع بيانات التاريخ يتكوَّن من عشر خانات مُقسَّمة إلى السنة، والشهر، واليوم يفصل بينها علامة الناقص (-) (DD-MM-YYYY)؛ بحيث يُقصَد بالـ (YYYY) السنة، و(MM) الشهر، و(DD) اليوم.

- الوقت (Time): نوع بيانات الوقت يتكون من ثماني خانات (على الأقل) تمثل الساعة، والدقيقة، والثانية وعلى هيئة (SS:MM:HH)؛ بحيث يُقصَد بـ (HH) الساعة (Hour)، و(MM) الدقيقة (Minute)، و(SS) الثانية (SeCond).

7-1-1-3 توصيف القيود في لغة الاستفسار البنائية والتعامل معها:

توفر لغة الاستفسار البنائية عدداً من أنواع القيود التي يمكِن توصيفها على قاعدة البيانات التي يجب على نظام إدارة قاعدة البيانات التأكُّد من تحققها على حالات قاعدة البيانات كافةً.

ويمكِن تقسيم أنواع القيود التي يمكن توصيفها بلغة الاستفسار البنائية إلى أربعة أنواع رئيسة، هي: قيود الحقول والمجال (Attribute and Domain Constraints)، قيود المفاتيح الرئيسة والسلامة المرجعية (Key and Referential Integrity Constraints)، وقيود السجلات (Tuple Constraints)، والقيود العامة على قاعدة البيانات (Assertions).

7-1-1-3-1 قيود الحقول والمجال (Attribute and Domain Constraints):

تسمح لغة الاستفسار البنائية بأن تأخذ قيمَ حقول الجداول القيمة الغائبة (Null) في حالة عدم إدخال قيم في هذه الحقول. وللتأكيد على وجوب إدخال قيم في حقلٍ ما مثل الاسم الأول أو اسم العائلة للموظفين، التي لا يمكن أن تكون قيمها غائبةً منطقياً؛ فإن لغة الاستفسار البنائية توفر قيد (Not Null) الذي يمكِن توصيفه على مثل هذه الحقول. ويعني هذا أن قِيم هذه الحقول يجب أن تتوفر في جميع سجلات الموظفين، ولا يمكن أن تكون غائبة.

توفر لغة الاستفسار البنائية أيضاً القيد «الفريد» (Unique) الذي يمكِن فرضه على أحد الحقول في حالة كانت قيمة هذا الحقل يجب أن تكون فريدةً لا تتكرر مع بقية السجلات في الجدول نفسه. فعلى سبيل المثال: يمكن فرض هذا القيد على رقم السجل المدني للموظفين (أو رقم بطاقة الأحوال المدنية) لكون قِيم مثل هذا الحقل لا يمكن أن تتكرر بين الموظفين. وعلى الرغم من أن قيمة الحقل المعرَّف على أساس أنه فريد، من المنطقي ألا تكون غائبة (Null)؛ فإن SQL تسمح بذلك. والسبب وراء ذلك يُعزَى إلى أن كلَّ قيمة غائبة تُعَدُّ مختلفةً عن قيمةٍ غائبةٍ أخرى للحقل نفسه؛ بمعنى أن القيمة الغائبة من حقل رقم الهاتف من سجل أحد الموظفين، على سبيل المثال، تُعَدُّ مختلفةً عن القيمة الغائبة من حقل رقم الهاتف من سجل موظف آخر.

أمَّا النوع الثالث من قيود الحقول التي توفرها لغة الاستفسار البنائية؛ فهو قيد المفتاح الرئيسي (Primary Key). ويعني هذا القيد عند توصيفه على أحد الحقول في جدولٍ ما، أن هذا الحقل هو المفتاح الرئيسي للجدول، كما سبق أن أشرنا لذلك سابقاً. وعند توصيف حقلٍ ما على أنه مفتاح رئيسي؛ فإن ذلك يعني ضمنياً أن الحقل فريدٌ ولا يمكن أن تكون قيمته غائبةً، بمعنى أن كلا القيدين السابقين يتحققان على الحقل عند تعريفه باعتباره مفتاحاً رئيسياً للجدول.

وفيما يلي الصيغة العامة لتوصيف قيود الحقول؛ بحيث يُقصَد بالعلامة (|) كلمة «أو» في حين يُقصَد بالكلمات الواقعة ضِمن أقواس مربعة بأنها كلمات اختيارية. ولكون كافة قيود الحقول تقع ضِمن قوسين مربعين؛ فإن هذا يعني أن توصيف قيودٍ على الحقول عمليةٌ اختيارية، وأن عدم وضع أيِّ قيدٍ من القيود الثلاثة أعلاه يعني أن الحقل قد تكون قيمته غائبة في بعض (أو كل) سجلاته، وأن قيم الحقل ليست فريدةً، وأن الحقل ليس مفتاحاً رئيسياً للجدول.

```
ColumnName     data-type     [[Not] Null | Unique | Primary Key]
```

وتوضِّح تعليمة الإنشاء التالية، المُستمَدَّة من قاعدة بيانات الجامعة الأهلية، قيد المفتاح الرئيسي وقيد القيم الغائبة (Not Null)؛ إذ يُبيِّن أن المفتاح الرئيسي لجدول المواد الدراسية هو رمز المادة الذي يتكون من سلسلة حرفية بمقاس ثابت طوله سبعة أحرف، كما يُوضِّح أن كلاً من حقل اسم المادة الدراسية (Title) وحقل عدد وحداتها الدراسية (Units) لا يمكن أن تكون قيمهما غائبتين.

```
CREATE TABLE COURSE_T
        (COURSE_ID      CHAR(7)        PRIMARY KEY,
         TITLE          CHAR(35)       NOT NULL,
         UNITS          NUMBER         NOT NULL);
```

كما يمكِن أيضاً فرضُ قيودٍ على مدى القيم التي من الممكن أن يأخذها حقلٌ ما باستخدام كلمة التحقق (CHECK). فعلى سبيل المثال: من الممكن أن يفرض قيد التحقق على عدد وحدات (أو ساعات) المادة الدراسية؛ بحيث يكون أكبر من صفر وأقل من ست وحدات دراسية، كما يوضِّحه المثال التالي:

```
CREATE TABLE COURSE_T
        (COURSE_ID    CHAR(7)      PRIMARY KEY,
         TITLE        CHAR(35)     NOT NULL,
         UNITS        NUMBER       NOT NULL CHECK (UNITS > 0 AND UNITS< 6));
```

1-1-3-1-1-7 إنشاء المجال (Domain Creation):

على الرغم من أنه يمكِن تعريف نوع البيانات لأيِّ حقل بشكلٍ مباشرٍ؛ فإنه يمكِن أيضاً تعريف مدى معين للقيم التي من الممكن أن يأخذها نوع البيانات الذي تمَّ تعريفه للحقل. وتُستخدَم عبارة إنشاء مجال (Domain Create)؛ لإنشاء نوعٍ جديد من البيانات مشتق من أحد أنواع البيانات الرئيسية التي توفرها لغة الاستفسار البنائية. وتُعَدُّ هذه الطريقة مفيدةً جداً عندما يشترك عددٌ من الحقول في نفس نوعية البيانات ومدى القيم. فعلى سبيل المثال: يمكِن تعريف رقم السجل المدني (أو رقم بطاقة الأحوال المدنية) على أنه مجال يتكون من نوع السلاسل الحرفية وأن طول سلسلته الحرفية مكوَّنٌ من عشرة حروف، كما يلي:

```
CREATE DOMAIN SoCial_IdentifiCation_Number AS
CHAR(10);
```

وعند توصيف أيِّ حقل يتعلق برقم السجل المدني يُستخدَم اسم المجال (SoCial_IdentifiCation_Number) عوضاً عن توصيفه باستخدام سلسلة حرفية مكوَّنة من عشرة أحرف (Char (10)). وتُعَدُّ هذه الطريقة في توصيف الحقول مفيدةً جداً؛ لكونها تسهِّل قراءة مكونات قاعدة البيانات من جانب، وإمكانية حصر التغييرات في مكانٍ واحد فقط من جانب آخر. فعلى سبيل المثال: يمكِن تغيير تعريف المجال الخاص برقم السجل المدني؛ ليصبح بطول اثني عشر حرفاً عوضاً عن عشرة أحرف، وسينعكس هذا التغيير على كلِّ الحقول التي تمَّ توصيف نوع بياناتها على أنه من نوع (SoCial_IdentifiCation_Number). وبناءً على ما سبق؛ فإن تعريف أيِّ مجال يتكون من جزأين: (1) نوع البيانات (وهو إجباري)، و(2) مدى القِيم (وهو اختياري).

ومثالٌ آخر: يمكِن تعريف مجال باسم «عدد الوحدات الدراسية» (CRS_UNITS)؛ بحيث يكون نوع بياناته هو الأعداد الصحيحة، وفي ذات الوقت، يفرض عليه قيد التحقق الذي ينصُّ على أن عدد الوحدات الدراسية يجب أن يكون أكبر من صفر وأقل من ست وحدات، الذي يمثل مدى القيم التي من الممكن أن يأخذها الحقل، كما يلي:

```
CREATE DOMAIN CRS_UNITS AS NUMBER
                    CHECK (CRS_UNITS > 0 AND CRS_UNITS< 6);
```

ويُمكِننا استخدامُ المجال الذي تمَّ إنشاؤه كنوعٍ لبيانات (Data Type) أيِّ حقل في قاعدة البيانات يُعرَف على أساس أنه من مجال CRS_UNITS، وبحيث يُطبق عليه قيد التحقق الذي ينصُّ على أن عدد وحدات المادة الدراسية يجب أن يكون أكبـر من صفر وأقل من ست وحدات. فمثلاً يمكن إعادة تعريف جدول المواد الدراسية؛ بحيث يستخدم المجال الذي تم تعريفه ليصبح كالتالي:

```
CREATE TABLE COURSE_T
        (COURSE_ID      CHAR(7)         PRIMARY KEY,
         TITLE          CHAR(35)        NOT NULL,
         UNITS          CRS_UNITS       NOT NULL);
```

أمَّا المثال التالي؛ فيوضِّح تعريف مجال الجنس (Gender) الذي وضع عليه قيد التحقق؛ بحيث تكون القيمة المُدخَلة لأيِّ حقل يستخدمه نوعاً لبياناته بأن تكون إمَّا ذكراً (Male) أو أنثى (Female)؛ وبحيث يستخدم الحرف الأول فقط من جنس الشخص (M) أو (F).

```
CREATE DOMAIN GENDER AS CHAR(1)
                    CHECK (VALUE IN ('M', 'F'));
```

تجدر الإشارة هنا إلى أن إنشاء مجال يُعَدُّ من ضِمن مقياس (SQL-92)؛ ولكنه ليس من الضرورة لنظام إدارة قاعدة البيانات أن يتبنَّى هذا الجزء من المقياس حتى يكون متوافقاً معه في المستوى المبـدئي أو المتوسط.

من الممكن أيضاً أن يتمَّ تعريف قيمة افتراضية لحقلٍ ما تُستخدَم من قِبل النظام في حالة عدم إدخال قيمة للحقل في أثناء عملية إدخال البـيانات. وتُستخدَم كلمة (DEFAULT) في مقياس (SQL-92) لتعريف القيمة الافتراضية الواجب إدخالها تلقائياً من قِبل نظام إدارة قاعدة البـيانات في حالة عدم إدخال قيمة للحقل. ويوضِّح المثال التالي طريقة استخدام القيمة الافتراضية في جدول

المواد الدراسية؛ بحيث تكون القيمة الافتراضية لعدد وحدات المادة الدراسية «ثلاثة» في حالة عدم إدخال قيمة لحقل عدد الوحدات الدراسية.

```
CREATE TABLE COURSE_T
        (COURSE_ID    CHAR(7)        PRIMARY KEY,
         TITLE        CHAR(35)       NOT NULL,
         UNITS        CRS_UNITS      NOT NULL DEFAULT 3);
```

7-1-1-3-2 قيود المفاتيح الرئيسية والسلامة المرجعية (Key and Referential Integrity Constraints):

تُستخدَم عبارة المفتاح الرئيسي (Primary Key) التي توفِّرها (SQL) لتعريف المفاتيح الرئيسية للجداول. أمَّا المفاتيح الخارجية؛ فيتمُّ تعريفها من خلال عبارة مفتاح خارجي (Foreign Key). وعلى الرغم من أن المفتاح الرئيسي يُمكِن تعريفه باعتباره قيداً على حقلٍ ما في الجدول، في أثناء تعريف الحقل، كما سبق إيضاح ذلك في مثال جدول المواد الدراسية أعلاه، غير أن المفتاح الرئيسي يجب أن يُعرَف بشكلٍ مستقل عن تعريف حقول الجدول عندما يكون المفتاح الرئيسي مكوناً من أكثر من حقل. ويوضِّح المثال التالي الذي يُعرِّف جدول المجموعات الدراسية في الجامعة الأهلية؛ أن المفتاح الرئيسي يتكوَّن من أربعة حقول هي: رمز المادة الدراسية (Course_ID)، ورقم الشعبة (أو المجموعة) (SeCtion_No)، والفصل الدراسي المنفَّذة فيه (Semester)، والسنة الدراسية المنفَّذة فيها (Year). كما يحتوي تعريف الجدول على مفتاحين خارجيين: الأول منهما يربط المجموعة الدراسية (أو الشعبة) بالمادة الدراسية التي تتبعها المجموعة (أو الشعبة) في جدول المواد الدراسية، أمَّا الثاني؛ فيربط المجموعة الدراسية (أو الشعبة) بعضو هيئة التدريس الذي يقوم بتدريسها في جدول أعضاء هيئة التدريس.

```
CREATE TABLE SECTION_T
        (COURSE_ID     CHAR(7)       NOT NULL,
         SECTION_NO    NUMBER        NOT NULL,
         SEMESTER      CHAR(10)      NOT NULL,
         YEAR          NUMBER        NOT NULL,
         FACULTY_ID    CHAR(8)       NOT NULL,
         LOCATION      CHAR(12)      NOT NULL,
        PRIMARY KEY (COURSE_ID, SECTION_NO, SEMESTER, YEAR),
        FOREIGN KEY (COURSE_ID) REFERENCES COURSE_T(COURSE_ID),
        FOREIGN KEY (FACULTY_ID) REFERENCES FACULTY_T(FACULTY_ID));
```

تجدر الملاحظة أنه ليس من الضروري أن يتطابق اسم الحقل باعتباره مفتاحاً خارجياً في جدولٍ ما مع اسم الحقل الذي يشير إليه في الجدول الآخر؛ ولكنه من الضروري أن يكون كلا الحقلين من نوعية البيانات نفسها (أو المجال نفسه). فمثلاً يمكِن تسمية حقل رمز عضو هيئة التدريس على أنه (FAC_ID) في جدول المجموعات الدراسية دون أن يغيِّر ذلك في الأمر من شيء ما دامت نوعية بيانات الحقل هي من نوعية بيانات الحقل (FaCulty_ID) نفسها في جدول أعضاء هيئة التدريس؛ وذلك كما يلي:

```
CREATE TABLE SECTION_T
        (COURSE_ID     CHAR(7)       NOT NULL,
         SECTION_NO    NUMBER        NOT NULL,
         SEMESTER      CHAR(10)      NOT NULL,
         YEAR          NUMBER        NOT NULL,
         FAC_ID        CHAR(8)       NOT NULL,
         LOCATION      CHAR(12)      NOT NULL,
         PRIMARY KEY (COURSE_ID, SECTION_NO, SEMESTER, YEAR),
         FOREIGN KEY (COURSE_ID) REFERENCES COURSE_T(COURSE_ID),
         FOREIGN KEY (FAC_ID) REFERENCES FACULTY_T(FACULTY_ID));
```

أمَّا في حالة تطابق مُسمَّى الحقلين في كلا الجدولين؛ فإنه بالإمكان الاستغناء عن ذِكر الحقل المشار إليه من قِبل المفتاح الخارجي كما يُوضِّح المثال التالي؛ حيث تمَّ الاستغناء عن ذكر اسم الحقل (Course_ID) والحقل (FaCulty_ID) عند تعريف كلا المفتاحين الخارجيين لكون مُسمَّياتهما في جدول المجموعات الدراسية متوافقةً مع مسمياتهما في جدول المواد الدراسية وجدول أعضاء هيئة التدريس، على التوالي:

```
CREATE TABLE SECTION_T
        (COURSE_ID    CHAR(7)      NOT NULL,
         SECTION_NO   NUMBER       NOT NULL,
         SEMESTER     CHAR(10)     NOT NULL,
         YEAR         NUMBER       NOT NULL,
         FACULTY_ID   CHAR(8)      NOT NULL,
         LOCATION     CHAR(12)     NOT NULL,
         PRIMARY KEY (COURSE_ID, SECTION_NO, SEMESTER, YEAR),
         FOREIGN KEY (COURSE_ID) REFERENCES COURSE_T,
         FOREIGN KEY (FACULTY_ID) REFERENCES FACULTY_T);
```

ويقومُ نظام إدارة قاعدة البيانات بإعطاء كلِّ قيد رمزاً يميِّزه عن بقية القيود المفروضة على قاعدة البيانات. غير أنه من المتعارف عليه عند إداريِّ قواعد البيانات إعطاء كلِّ قيد مُسمَّاه الخاص الذي يساعد على فهمهم لطبيعة القيد (سواءٌ أكان مفتاحاً رئيسياً أم خارجياً أم غير ذلك)، ومجال تطبيقه (سواءٌ أكان على حقل أم جدول أم قيد عامٍ). كما يساعدهم ذلك على التعرُّف على القيود المختلفة وتعطيل العمل بها أو تعديلها. ويوضِّح المثال التالي إحدى الطرق المتَّبعة عند تسمية القيود؛ بحيث تم إدراج اسم الجدول ضِمن مُسمَّى القيد وطبيعة كونه مفتاحاً رئيسياً (PK) أو خارجياً (FK).

```
CREATE TABLE SECTION_T
        (COURSE_ID    CHAR(7)      NOT NULL,
         SECTION_NO   NUMBER       NOT NULL,
         SEMESTER     CHAR(10)     NOT NULL,
         YEAR         NUMBER       NOT NULL,
         FACULTY_ID   CHAR(8)      NOT NULL,
         LOCATION     CHAR(12)     NOT NULL,
CONSTRAINT SECTION_PK PRIMARY KEY (COURSE_ID, SECTION_NO, SEMESTER, YEAR),
CONSTRAINT SECTION_FK1 FOREIGN KEY (COURSE_ID)
                                   REFERENCES COURSE_T(COURSE_ID),
CONSTRAINT SECTION_FK2 FOREIGN KEY (FACULTY_ID)
                                   REFERENCES FACULTY_T(FACULTY_ID));
```

ولكون المفاتيح الخارجية هي الطريقة الوحيدة لتمثيل العلاقات (أو الارتباطات) بين الحالات المدوَّنة في الجداول العلاقية؛ فإنها تمثِّل قيوداً للسلامة المرجعية. فعلى سبيل المثال: عندما نقول: إن «كلَّ مجموعة دراسية تتبع لمادة دراسية واحدة فقط»؛ فإن هذا يُعَدُّ قيداً بين المجموعة الدراسية والمادة الدراسية التي تتبعها المجموعة. وتُمثَّل هذه العلاقة من خلال تعريف

مفتاح خارجي في جدول المجموعات الدراسية يشير إلى المفتاح الرئيسي في جدول المواد الدراسية، كما أسلفنا في المثال أعلاه. غير أن السؤال المتعلق بقيد السلامة المرجعية هو ماذا يحصل لو تغير رمز المادة الدراسية التي تتبعها إحدى المجموعات الدراسية، أو حذفت مادة دراسية من جدول المواد الدراسية، وكان يتبعها عددٌ من المجموعات في جدول المجموعات الدراسية؟ في هذه الحالة، ما مصير المفاتيح الخارجية؟ إلامَ تشيرُ هذه المفاتيح؟

كذلك هو الحال عند إضافة مجموعة دراسية دون تحديد لرمز المادة الدراسية التي تتبعها المجموعة الدراسية، أو تمَّ إدخال المجموعة الدراسية بـقيمة لرمز مادة دراسية غير موجودة أساساً في جدول المواد الدراسية. ما نتيجة مثل هذه العمليات؟

إن ردَّة الفعل الافتراضية في نظم قواعد البيانات العلاقية حسب مقياس (SQL-92)؛ هو رفض مثل هذه العمليات دون تغيير لمحتويات جداول قاعدة البـيانات؛ وذلك لكونها تؤدي إلى اختراق قيود السلامة المرجعية. وتُسمَّى ردَّة الفعل هذه بعملية مَنْع أو إيقاف التنفيذ (RestriCt). ولكن مقياس (SQL-92) يوفِّر ثلاثة بدائل أخرى لمصمِّمي قواعد البيانات؛ إضافةً إلى رَدَّة الفعل الافتراضية، تمكِّنهم من اختيار الفعل المناسب عند اختراق قيود السلامة المرجعية حسب الوضع الذي يتناسب مع قواعد بـياناتهم. ويتمُّ ذلك من خلال ربط الفعل المناسب بـقيد المفتاح الخارجي. أمَّا ردود الفعل الثلاثة؛ فهي: وَضْع المفتاح الخارجي بحيث تكون قيمته غائبة (Set Null)، التغيير المتسلسل (CasCade)، وَضْع المفتاح الخارجي في الحالة الافتراضية (Set Default). وعند اختيار أحد ردود الفعل المناسبة؛ فإنه يجب أن يرتبط بالفعل نفسه، سواء كان فعل تعديل (On Update) أم فعل حذف (On Delete). ولقد سبق شرح مفاهيم قيود السلامة المرجعية في الجزء (4-1-1-4-4)، والتعامل مع اختراقاتها من قِبل نظام إدارة قاعدة البيانات في الجزء (4-1-1-4-5).

ويُوضِّح المثال التالي نوعين من ردود الفعل: الأول منهما ينصُّ على أن تحديث رمز المادة الدراسية في جدول المواد الدراسية يجب أن ينعكس على (أو يتسلسل إلى) جميع مجموعات المادة الدراسية في جدول المجموعات الدراسية، وكذلك هو الحال بالنسبة لتحديث رمز عضو هيئة التدريس الذي يجب أن ينعكس على (أو يتسلسل إلى) جميع السجلات التي يُدَرِّسُها عضو هيئة التدريس نفسه في جدول المجموعات الدراسية. أما رَدُّ الفعل الثاني فيتمثل في حالة حذف سجل أحد

أعضاء هيئة التدريس من جدول أعضاء هيئة التدريس. في هذه الحالة سيتمُّ تغيير قيمة حقل رمز عضو هيئة التدريس؛ بحيث يأخذ القيمة الافتراضية وهي (No Body) في كلِّ سجل من سجلات جدول المجموعات الدراسية التي تحتوي على قيمة رمز عضو هيئة التدريس الذي تمَّ حذفه من جدول أعضاء هيئة التدريس. ومن الجدير ذِكره أنه يتوجب في هذه الحالة أن يكون من ضِمن سجلات أعضاء هيئة التدريس سجلٌ يمثل الحالة الافتراضية وهي "No Body"؛ وبحيث يكون مفتاحها الرئيسي هو "No Body". ويعني هذا أنه كلما تم حذف سجل لعضو هيئة تدريس من جدول أعضاء هيئة التدريس؛ فإن كافة المجموعات الدراسية المُسنَد تدريسها لعضو هيئة التدريس الذي تم حذف سجله ستصبح مرتبطةً بالسجل الافتراضي في جدول أعضاء هيئة التدريس ذي المفتاح الرئيسي "No Body".

```
CREATE TABLE SECTION_T
        (COURSE_ID      CHAR(7)         NOT NULL,
         SECTION_NO     NUMBER          NOT NULL,
         SEMESTER       CHAR(10)        NOT NULL,
         YEAR           NUMBER          NOT NULL,
         FACULTY_ID     CHAR(8)         NOT NULL DEFAULT 'No Body',
         LOCATION       CHAR(12)        NOT NULL,
CONSTRAINT SECTION_PK PRIMARY KEY (COURSE_ID, SECTION_NO, SEMESTER, YEAR),
CONSTRAINT SECTION_FK1 FOREIGN KEY (COURSE_ID)
                                REFERENCES COURSE_T(COURSE_ID)
                                        ON UPDATE CASCADE,
CONSTRAINT SECTION_FK2 FOREIGN KEY (FACULTY_ID)
                                REFERENCES FACULTY_T(FACULTY_ID)
                                        ON DELETE SET DEFAULT
                                        ON UPDATE CASCADE);
```

7-1-1-3-3 قيود السجلات (Tuple Constraints):

إضافةً إلى القيودِ التي من الممكن أن تُفرَض على الحقول والقيود التي تُفرَض لتأكيد السلامة المرجعية، توفر (SQL) قيود السجلات التي من الممكن أن تُفرَض على أكثر من حقل في الجدول نفسه وفي الوقت نفسه. ولتعريف مثل هذه القيود تُستخدَم الكلمة (CHECK) في نهاية تعريف الجدول. ويُسمَّى هذا النوع من القيود بقيود السجلات؛ لأنه يُفرَض على كلِّ سجل على حِدَة بشكلٍ منفرد عند إضافة السجل أو التعديل عليه. فعلى سبيل المثال: لا يمكن أن تكون مادة دراسية ما متطلباً دراسياً للمادة نفسها (وإلا لما كان بإمكان أيِّ طالب من التسجيل في المادة الدراسية).

ولتعريف مثل هذا القيد نستخدم كلمة التحقق (CHECK) في نهاية تعريفنا لجدول المواد الدراسية المتطلبة (Prerequisite_T)، كما يلي:

```
CREATE TABLE PREREQUISITE_T
        (COURSE_ID              CHAR(7)         NOT NULL,
         PREREQUISITE_ID        CHAR(7)         NOT NULL,
         CONSTRAINT RRFREQUISITE_PK PRIMARY KEY (COURSE_ID, PREREQUISITE_ID),
         CONSTRAINT PREREQUISITE_FK1 FOREIGN KEY (COURSE_ID)
                                        REFERENCES COURSE_T(COURSE_ID),
         CONSTRAINT PREREQUISITE_FK2 FOREIGN KEY (PREREQUISITE_ID)
                                        REFERENCES COURSE_T(COURSE_ID)
         CHECK (COURSE_ID <> PREREQUISITE_ID));
```

ويُلاحَظ أن القيد «لا يمكِن أن تكون أية مادة دراسية متطلباً دراسياً للمادة نفسها» عبارة عن قيدٍ يتعلق بـأكثر من حقل ضِمن السجل نفسه. وعند إدخال سجل جديد لجدول المواد الدراسية المتطلَّبة أو تعديل أيِّ سجل موجود فيه، يقوم نظام إدارة قاعدة البيانات بـالتأكُّد من تحقق هذا القيد. وفي حالة خَرْق هذا القيد من قِبل عملية تعديل لسجل موجود أو إدخال لسجل جديد؛ فإن نظام إدارة قاعدة البيانات سيقوم بـرفض تنفيذ العملية.

4-3-1-1-7 قيود عامة (Assertions):

القيود العامة هي قيود قد تتعلق بـأكثر من جدول في الوقت نفسه. لذا؛ فإن هذا النوع من القيود يجب أن ينطبق على أيِّ حالة تكون فيها قاعدة البيانات عوضاً عن حالة الجدول كما هو الحال بالنسبة لقيود السجلات، أو حالة الحقل كما هو الحال بالنسبة لقيود الحقول. وتُستخدَم عبارة (CREATE ASSERTION) لتعريف القيود العامة. والشكل العام للقيد العام، كما يلي:

```
CREATE ASSERTION Assertion_Name CHECK condition;
```

وعلى الرغم من أن القيود العامة تتطلَّب التعرُّف على القوة الحقيقية للاستفسارات في لغة الاستفسار البنائية؛ فإننا نقدِّم المثال التالي، الذي سيتضح معناه أكثر عند استعراض الاستفسارات في لغة الاستفسار البـنائية. إنَّ القيد العام الذي يهدف إليه المثال التالي هو التحقق من أن أيَّ طالب في الجامعة الأهلية لا يمكن أن يُسجَّل (Registers) في مواد دراسية يفوق عدد ساعاتها

الإجمالية أكثر من اثنتي عشرة وحدة دراسية في أيِّ فصل كان من أية سنة دراسية. وللتأكُّد من تحقق هذا القيد على جميع حالات قاعدة البيانات؛ فإن ذلك يتطلب الرجوع لبيانات حقول تتبع لأكثر من جدول. ويقع على نظام إدارة قاعدة البيانات التأكُّد من أن هذا القيد، حسب تعريفه التالي، متحقق في جميع حالات قاعدة البيانات.

```
CREATE ASSERTION REGISTRATION _CHECK
CHECK (Not Exists (
        (Select Sum(units)
         From Student_T s, Course_T c, Section_T t, Enrollment_T e
         Where s.Student_ID = e.Student_ID AND
          e.Course_ID = t.Course_ID AND
          e.Section_No = t.Section_No AND
          e.Semester = t.Semester AND
          e.Year = t.Year AND
          c.Course_ID = t.Course_ID
          Group By s.student_ID, e.year, e.semester) > 12);
```

وتجدر الإشارة هنا إلى بطء تنفيذ عمليات التعديل؛ سواء من خلال عمليات الإضافة، أو الحذف، أو التحديث على قاعدة البيانات عند استخدام القيود العامة؛ وذلك لضرورة مراجعة نظام إدارة قاعدة البيانات للقيود العامة في كلِّ مرة يتمُّ التعديل على قاعدة البيانات.

7-1-1-3-5 تعديل القيود والتحكم في تطبيقها:

7-1-1-3-5-1 تعديل القيود:

من الممكن إضافة، أو تعديل، أو إزالة القيود في أيِّ وقتٍ كان. وتعتمدُ طريقة التعديل حسب القيد نفسه؛ بمعنى: إن كان متعلقاً بحقل، أو جدول، أو قاعدة البيانات. وكما أسلفنا سابقاً؛ إنه من الضروري إعطاء القيود مُسمَّيات معيَّنة حتى يمكن التعرُّف عليها ومن ثم تعديلها عند الرغبة في ذلك.

7-1-1-3-5-2 إزالة القيود:

لإزالة قيدٍ ما من جدول تُستخدَم عبارة «إزالة قيد» (Drop Constraint) بالإضافة لعبارة «تعديل جدول» (Alter Table) كما هو مُوضَّح في الشكل العام التالي للتعليمة:

```
ALTER TABLE Table_Name DROP CONSTRAINT Constraint_Name;
```

فعلى سبيل المثال: لو أردنا إزالة المفتاح الخارجي من جدول المجموعات الدراسية، والذي عُرف على أساس أنه قيد جدول، والمُسمَّى (SECTION_FK2) الذي يشير إلى أعضاء هيئة التدريس المكلَّفين بتدريس المجموعات الدراسية في جدول أعضاء هيئة التدريس كما هو مُبيَّن في تعريف جدول المجموعات الدراسية التالي:

```
CREATE TABLE SECTION_T
        (COURSE_ID      CHAR(7)         NOT NULL,
         SECTION_NO     NUMBER          NOT NULL,
         SEMESTER       CHAR(10)        NOT NULL CONSTRAINT SEM_NAME CHECK
                                             (VALUE IN 'FALL', 'SPRING', 'SUMMER'),
         YEAR           NUMBER          NOT NULL,
         FACULTY_ID     CHAR(8)         NOT NULL,
         LOCATION       CHAR(12)        NOT NULL,
CONSTRAINT SECTION_PK PRIMARY KEY (COURSE_ID, SECTION_NO, SEMESTER, YEAR),
CONSTRAINT SECTION_FK1 FOREIGN KEY (COURSE_ID)
                        REFERENCES COURSE_T(COURSE_ID),
CONSTRAINT SECTION_FK2 FOREIGN KEY (FACULTY_ID)
                        REFERENCES FACULTY_T(FACULTY_ID));
```

فإنه يمكِن استخدام عبارة إزالة قيد السلامة المرجعية المذكور أعلاه، كما يلي:

```
ALTER TABLE SECTION_T DROP CONSTRAINT SECTION_FK2;
```

كما يمكِن إزالة القيد المفروض على حقل الفصل الدراسي (Semester)، الذي عُرف على أساس أنه قيد حقل؛ بحيث يجب أن تكون قيمته إمَّا الخريف (Fall)، أو الربيع (Spring)، أو الصيف (Summer)، كما يلي:

```
ALTER TABLE SECTION_T DROP CONSTRAINT SEM_NAME;
```

7-1-1-3-5-3 إضافة القيود:

لإضافة القيود تُستخدَم عبارة «إضافة قيد» (ADD CONSTRAINT). ولو أردنا إعادة تعريف قيد السلامة المرجعية الذي تمَّت إزالته، على سبيل المثال، فإنه يمكن إضافته مجدداً، كما يلي:

```
ALTER TABLE SECTION_T ADD CONSTRAINT SECTION_FK2 FOREIGN KEY (Faculty_ID)
                                        REFERENCES FACULTY_T(Faculty_ID);
```

كما يمكِن إضافة القيد المفروض على حقل الفصل الدراسي، في جدول المجموعات الدراسية، بعد إزالته؛ بحيث يمكن أن تكون قيمته واحدة من أربع قيم عوضاً عن ثلاثٍ، وذلك من خلال إضافة فصل الشتاء (Winter) الذي من الممكن أن تُنفِّذ فيه الجامعة بعضاً من موادها الدراسية، كما يلي:

```
ALTER TABLE SECTION_T ADD CONSTRAINT SEM_NAME
        CHECK (SEMESTER IN ('FALL', 'SPRING', 'SUMMER', 'WINTER'));
```

ويُلاحَظ في تعريفنا للقيد السابق أنه أصبح قيد جدول عوضاً عن قيد حقل؛ وذلك لكون لغة الاستفسار البنائية لا توفر طريقة لتعريف قيود حقول بعد إنشاء الجداول.

7-1-1-3-5-4 تعطيل عمل القيود واستعادة العمل بها:

يُمكِن تعطيل العمل بأيٍّ من القيود مع الاحتفاظ بها وإعادة عملها مرةً أخرى لاحقاً. ولتعطيل العمل بقيدٍ ما؛ تُستخدَم عبارة «تعطيل القيد» (Disable Constraint). أما عبارة «إعادة العمل بالقيد» (Enable Constraint) فتُستخدَم لإعادة العمل بالقيد. وتختلف هاتان العمليتان عن عمليتَيْ حذف القيود وإعادة تعريفها لكون هاتين العمليتين لا تُلغيان قيوداً معرفة أو تقومان بتعريف قيود جديدة، وإنما تُستخدَمان لإيقاف العمل بالقيود لفترةٍ ما، ومن ثم إعادة العمل بها مرةً أخرى. ويمكِن تعطيل عمل قيد السلامة المرجعية المتعلق بأعضاء هيئة التدريس في جدول المجموعات الدراسية، كما يلي:

```
ALTER TABLE SECTION_T DISABLE CONSTRAINT SECTION_FK2;
```

ويمكن استعادة العمل بالقيد، كما يلي:

```
ALTER TABLE SECTION_T ENABLE CONSTRAINT SECTION_FK2;
```

ويُستَفاد من العمليتين السابقتين بشكلٍ خاص للحالات الاستثنائية التي لا تتماشى مع القواعد العامة المعمول بها في منظمةٍ ما، وبالتالي عدم التمكُّن من تدوينها في قاعدة البيانات؛ بسبب القيود المفروضة على قاعدة البيانات. فعلى سبيل المثال: قد يُسمَح لطالب ما بالتسجيل، في فصلٍ دراسيٍّ ما، بساعات دراسية يقل عددها (أو يزيد) عن عدد الساعات الدراسية المسموح به نظاماً؛ نظراً لأن الفصل الدراسي الذي سيقوم الطالب بالتسجيل فيه هو الفصل الذي من المتوقع ان يتخرَّج فيه الطالب من الجامعة. ومثالٌ آخر هو عندما يُسمَح لموظفٍ ما بالتمتع بإجازة سنوية يزيد عدد أيامها عن الفترة المسموح بها نظاماً لسبب طارئ لديه. في مثل هذه الحالات الاستثنائية؛ يمكِن استخدام تعليمة تعطيل العمل بالقيود، وبعد تدوين بيانات الحالة المخالفة للقيد؛ يتمُّ إعادة العمل بالقيد من جديد. ويعني هذا أن قاعدة البيانات قد تحتوي على بيانات تبدو مخالفةً لقواعد العمل؛ ولكنها في الواقع ليست كذلك؛ طالما أن البيانات قد تمَّ تدوينها بموافقةٍ من قِبل صاحب الصلاحية المخوَّل بذلك وفقاً للنظام المعمول به في المنظمة.

ولتعطيل العمل بالقيود وإعادة العمل بها فائدةٌ أخرى تتمثل عند نقل كميات كبيرة من البيانات من قاعدة بيانات إلى قاعدة بيانات أخرى؛ نظراً لأن البيانات المدوَّنة في قاعدة البيانات التي سيتمُّ نقل البيانات منها سليمة؛ من حيث القيود المفروضة عليها، وبالتالي لا داعي للتحقق منها عند عملية نقل البيانات لقاعدة البيانات الأخرى. ولذلك؛ فإنه من المفيد جداً، من حيث السرعة في عملية نقل البيانات، تعطيل العمل بالقيود في قاعدة البيانات التي سيتمُّ نقل البيانات إليها؛ كونها ستوفر الوقت اللازم للتحقق من سلامة البيانات على بيانات هي في الأصل سليمة. وبعد إجراء عملية نقل البيانات يُعَاد العمل بالقيود على قاعدة البيانات التي تمَّ نقل البيانات إليها. وبذلك تتمُّ عملية نقل البيانات بشكلٍ أسرع مما هي عليه لو لم يتم تعطيل العمل بالقيود خاصة إذا ما كان حجم البيانات المنقولة كبيراً جداً وعدد القيود المفروضة على قاعدة البيانات كبير أيضاً.

7-1-1-3-5-5 تأخير العمل بالقيود (Deferring the CheCking of Constraints):

إن التمكُّن من تعطيل العمل بالقيود هو أمرٌ مهم جداً؛ كونه يمكِّن من تدوين حالات استثنائية في قاعدة البيانات لا تنطبق عليها (بعض) القيود المفروضة على قاعدة البيانات. إضافةً إلى ذلك؛ فإن تعطيل العمل بالقيود من شأنه تحسين أداء النظام في حال نقل بيانات كبيرة من قاعدة بيانات إلى أخرى. ومع ذلك؛ فإن تعطيل العمل بالقيود لا يمكِّن من تحسين أداء النظام، بالشكلِ المناسب، في حال كَثُرت عمليات التراجع (أو الانسحاب) عن التعديلات التي تُجرَى على قاعدة البيانات، أو من التغلُّب على مشكلة ما يُعرف بالقيود المرتبطة بشكل حلقي (أو القيود الحلقية، اختصاراً) بالشكل الأمثل (Al-Houmaily, 2013; GarCia-Molina et al, 2014). ولذلك توفر لغة الاستفسار البنائية طريقة «تأخير العمل بالقيود» (Deferred ConsistenCy Constraints).

إن عملية تأخير العمل بالقيود تعني تأخير التحقق منها إلى وقتٍ معين. وهذا الوقت هو الوقت الذي يُتِم فيه المستفيد تعامله مع قاعدة البيانات. ويقودنا هذا إلى التعرف على «المعاملات» (TransaCtions)، في نظم قواعد البيانات، التي تمثل الوسيلة الوحيدة للتعامل مع البيانات. والمعاملة، في نظم قواعد البيانات، هي برنامج حاسوبي يتعامل في بعض أجزائه مع قاعدة البيانات. ويُمكِن أن تنتهي المعاملة الواحدة بإحدى طريقتين: إما أن تنتهي بشكلٍ كاملٍ وتنعكس نتائجها كافة على قاعدة البيانات، أو أن تفشل في أثناء عملية تنفيذها؛ بسبب الرغبة في التراجع (أو الانسحاب) من عملية تنفيذها، أو لأيِّ سبب آخر، ولا ينعكس أيٌّ من نتائجها على قاعدة البيانات. ولكلِّ معاملة بداية ونهاية. فبداية المعاملة قد تكون ضمنية؛ تتمثل في أول تعامل لها مع قاعدة البيانات، أو أن تكون صريحة؛ تتمثل في عبارة تدل على بدايتها مثل تعليمة «البداية» (Begin). كما أن لكل معاملة نهاية تتمثل في تعليمة تدل على الرغبة في «تثبيت النتائج» (Commit)، أو الرغبة في «التراجع» (أو «الانسحاب») (Abort (or RollbaCk)). ومن خلال هاتين التعليمتين تتمكَّن قاعدة البيانات من معرفة بداية ونهاية كلِّ معاملة على حِدَة؛ الأمر الذي يُمكِّن نظام إدارة قاعدة البيانات من تأخير عملية التحقق من القيود المفروضة على بياناتها إلى حين انتهاء المعاملة عوضاً عن التحقق منها عند انتهائها من تنفيذ كلِّ عملية على حِدة.

7-1-1-3-5-5-1 تأخير العمل بالقيود، وأداء نظام إدارة قاعدة البيانات:

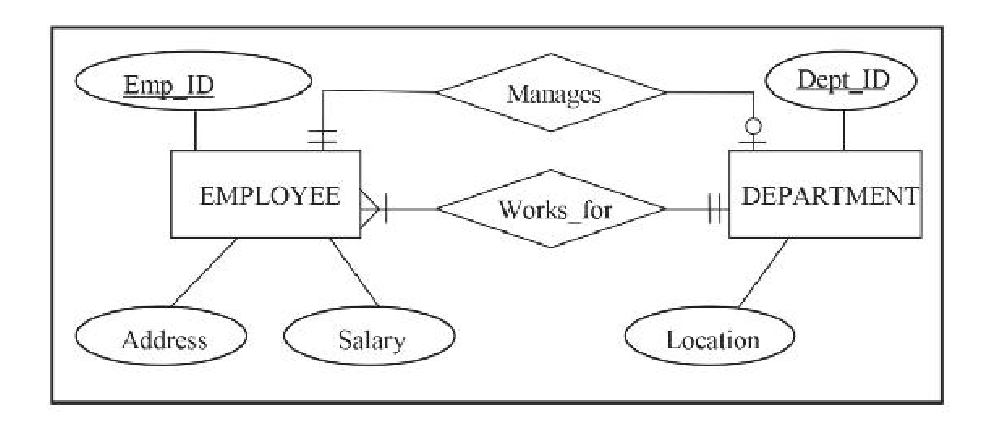
عند استخدام طريقة تأخير العمل بالقيود؛ فإن جميع الموارد اللازمة للتحقق من القيود يتمُّ الاستحواذ عليها عند انتهاء المعاملة من التنفيذ واستقبال نظام إدارة قاعدة البيانات لتعليمة تثبيت المعاملة. على النقيض من ذلك؛ فإن جميع الموارد اللازمة للتحقق من القيود عندما تكون «فورية» (Immediate) يتمُّ الاستحواذ عليها عند تنفيذ كل تعليمة تعديل تتضمَّنها المعاملة على حِدَة. ويعني هذا أن هنالك عملية «موازنة» (trade-off) بين الطريقتين. فطريقة تأخير العمل بالقيود تمكِّن من الاستحواذ على الموارد اللازمة للتحقق من القيود فترةً أقل من قرينتها الفورية؛ سامحةً بذلك لمزيدٍ من التزامن في الاستحواذ على الموارد من قِبل المعاملات المنفَّذة على النظام، وفي نفس الوقت عدم تضييع أيِّ وقت للتحقق من القيود إلى حين انتهاء تنفيذ كل معاملة. أمَّا من الناحية الأخرى؛ فإن تأخير العمل بالقيود قد يؤدي إلى تنازع المعاملات المنفذة على النظام على الموارد التي تتطلَّبها عملية التحقق من القيود عند انتهاء تنفيذ المعاملات وإصدارها لتعليمات تثبيتها، الأمر الذي قد يؤدي إلى كثرة المعاملات التي يتوجَّب إفشال تنفيذها؛ بسبب هذا التنازع على الموارد المطلوبة للتحقق من القيود، وبالتالي ضياع الوقت الذي قضته هذه المعاملات لتنفيذها وانعكاسه على أداء النظام (Al-Houmaily, 2013).

وبناءً على عملية الموازنة بين الطريقتين أعلاه؛ يمكن القول بأن طريقة تأخير العمل بالقيود من شأنها تحسين أداء النظام بشكلٍ كبيرٍ في حال كثرت المعاملات التي تستدعي التراجع بناءً على رغبات المستفيدين أو الفشل بسبب اكتظاظها على موارد النظام، في أثناء تنفيذها، الأمر الذي يؤدي إلى فشلها. خلاف ذلك؛ فإن الطريقة الفورية للتحقق من القيود هي الأجدى.

7-1-1-3-5-5-2 تأخير العمل بالقيود و «القيود الحلقية» (CyCliC ConsistenCy Constraints):

لإيضاح مشكلة القيود الحلقية (Al-Houmaily, 2013; GarCia-Molina et al, 2014) وأهمية تأخير العمل بالقيود لحَلِّها؛ لنفترض المثال الموضَّح في الشكل (7-2) (Al-Houmaily, 2013) الذي يحتوي على كينونتين: كينونة «الموظف» (EMPLOYEE)، وكينونة «القسم» (DEPARTMENT). كما يحتوي الشكل على علاقتين: العلاقة الأولى هي علاقة «يعمل في» (Works_for) وتنصُّ على أن «كلَّ موظف يعمل في قسم واحد وواحد فقط، وكل قسم يعمل فيه واحد أو أكثر من الموظفين». أمَّا العلاقة الثانية؛ فهي علاقة «يدير»

(Manages) وتنصُّ على أن «كل موظف يدير صفر أو واحد من الأقسام، وكل قسم يُدار من قِبل واحد وواحد فقط من الموظفين».



شكل رقم (7–2): مثال لمخطط مفاهيمي يحتوي على «قيود حلقية».

عند تحويل المخطط المفاهيمي في الشكل (7-2) إلى النموذج العلاقي؛ يتمُّ إنشاء جدولين (كما سبق إيضاحه في الفصل الخامس): الأول منهما يقابل كينونة الموظف، والثاني يقابل كينونة القسم. كما يتوجَّب عند عملية التحويل ملاحظة قيود التعددية؛ بحيث يتوجب أن يرتبط كلُّ موظف بقسم، وكل قسم بموظف يديره. ويمكن ترجمة قيود التعددية هذه إلى النموذج العلاقي من خلال وَضْع القيد (NOT NULL) على حقل (Dept_ID) في جدول «الموظف» الذي يدل على رقم القسم الذي يعمل فيه كلُّ موظف، ووضع القيد (NOT NULL) على حقل (Mgr_ID) في جدول «القسم» الذي يدل على رقم الموظف الذي يدير كل قسم.

تقودنا عملية التحويل هذه من النموذج المفاهيمي إلى النموذج العلاقي لمجموعة تعليمات SQL كما هو في الشكل رقم (7-3). ويُلاحَظ في الشكل أنه قد تمَّ إدراج تعليمات إنشاء قيود السلامة المرجعية لكلا الجدولين بعد إدراج تعليمات إنشاء الجدولين. والسبب في ذلك أن قاعدة البيانات ستمنعنا من إنشاء أيٍّ من الجدولين في حال ضُمِّن قيد السلامة المرجعية الخاص به؛ لكون الجدول المشار إليه من قِبل القيد لم يتم إنشاؤه بعد، وبالتالي عدم التمكن من التعرُّف عليه من قِبل نظام إدارة قاعدة البيانات.

```
CREATE TABLE EMPLOYEE_T
(Emp_ID         INTEGER        PRIMARY KEY,
...
...
Dept_ID         INTEGER        NOT NULL);

CREATE TABLE DEPARMENT_T
(Dept_ID        INTEGER        PRIMARY KEY,
...
...
Mgr_ID          INTEGER        NOT NULL);

ALTER TABLE EMPLOYEE_T ADD CONSTRAINT Emp_FK FOREIGN KEY (Dept_ID)
    REFERENCES DEPARTMENT_T(Dept_ID) DEFERRABLE INITIALLY IMMEDIATE;

ALTER TABLE DEPARTMENT_T ADD CONSTRAINT Dept_FK FOREIGN KEY (Mgr_ID)
REFERENCES EMPLOYEE_T(Emp_ID) DEFERRABLE INITIALLY IMMEDIATE;
```

شكل رقم (7–3): تعليمات SQL لإنشاء جدولين يحتويان على «قيود حلقية».

بناءً على تعليمات إنشاء الجدولين في الشكل (7-3)، لنفترض إحدى المعاملات التي تتضمَّن تعليمتين: إحداهما لإضافة سجل في جدول «الموظف»، وأخرى لإضافة سجل في جدول «القسم»، كما هو مُوضَّح في الشكل رقم (7-4). عند بداية تنفيذ هذه المعاملة؛ فإن نظام إدارة قاعدة البيانات سيقوم بإفشال تنفيذ المعاملة (بغض النظر عن ترتيبنا للعمليتين اللتين تحتويهما). ويُعزَى السبب وراء ذلك إلى أنه لا يمكن إضافة سجل في جدول الموظف؛ بحيث يكون حقل «رمزالقسم» (Dept_ID) الذي يعمل فيه الموظف ذا قيمةٍ غائبة (NULL)، أو أن يُضاف سجل في جدول القسم؛ بحيث يكون رقم الموظف الذي «يديره» (Mgr_ID) ذا قيمةٍ غائبة (NULL). ليس هذا فحسب؛ ولكن حتى لو وضعنا قيماً ما في كلا السجلين (كما هو مُوضَّح في الشكل)؛ فإن تنفيذ المعاملة سيفشل أيضاً بمجرد محاولة تنفيذها. ويرجع السبب وراء ذلك إلى أن كل جدول من الجدولين يشير (بحسب قيود السلامة المرجعية المفروضة عليه) إلى الجدول الآخر. ويعني هذا أن إضافة أيّ سجل في أحد الجدولين يُحتِّم وجود سجلٍ مقابل في الجدول الآخر. لذا؛ فإنه يتعذر إضافة أي سجل لأيٍّ من الجدولين؛ بسبب هذين القيدين الحلقيين. وقد تمتدُّ القيود الحلقية، بشكلٍ عام، لتشمل أكثر من جدولين.

```
INSERT INTO Employee_T values(100,..,1);
INSERT INTO Department_T values(1,..,100);
COMMIT;
```

شكل رقم (7–4): معاملة لإضافة سجلين: أحدهما في جدول «الموظف»، والآخر في جدول «القسم».

إنَّ الطريقة الوحيدة لحلِّ المعضلة أعلاه؛ هي من خلال كسر حلقة القيود المفروضة على الجدولين؛ بحيث يتمُّ تأخير التحقق من أيٍّ منهما (أو كلاهما) حتى انتهاء المعاملة وإرسالها لتعليمة التثبيت. وهذه هي الفائدة الثانية من طريقة تأخير العمل بـالقيود. ويعني هذا أن طريقة تأخير العمل بالقيود تسمح بخرق القيود المفروضة على قاعدة البيانات بشكلٍ مؤقتٍ من قِبل أيَّة معاملة إلى حين انتهاء تنفيذ كافة التعليمات التي تحتويها المعاملة. وعندئذٍ فقط؛ يتمُّ التحقق من سلامة القيود المفروضة على قاعدة البيانات. ويُمكِن ذلك طالما أن القيود التي يُرْغَب في تأخير العمل بها قد تمَّ تعريفها؛ بحيث تكون قابلةً للتأخير، كما هو مُوضَّح في الشكل (7-3). ولتأخير العمل بقيدٍ ما إلى حين وقت تثبيت نتائج المعاملات أو لإرجاعه لشكله الفوري، توفر لغة الاستفسار البنائية التعليمة التالية:

```
SET CONSTRAINTS ConstName DEFERRED | IMMEDIATE
```

بمجرد وَضْع أحد القيدين في وضع مؤخر، في مثالنا السابق، سيتمُّ كسر حلقة القيود والتمكُّن من تنفيذ المعاملة من خلال إضافة سجل لجدول الموظف، ومن ثم سجل لجدول القسم، كما هو مُوضَّح في الشكل رقم (7-5). وعند الانتهاء من تنفيذ كلتا عمليتي الإضافة، وإصدار تعليمة التثبيت؛ سيقوم نظام إدارة قاعدة البيانات من التحقق من كافة القيود التي قامت المعاملة بـتأخير التحقق منها. وفي حال كانت البيانات المُدخَلة سليمةً؛ سيتمُّ تثبيت نتائج المعاملة. أمَّا إذا لم تكن البيانات كذلك، سيتمُّ إفشال تنفيذ المعاملة والتراجع عمَّا قامت به من تعديلات على قاعدة البيانات. وفي مثالنا، سيتمُّ التحقق من حقل القسم الدراسي في جدول «الموظف» فقط؛ وذلك لكون المعاملة قد أضافت فعلياً سجلاً في جدول «القسم» يحقق قيد السلامة المرجعية المفروض على الجدول قبل وصول المعاملة لنهايتها وإصدارها لتعليمة التثبيت، وأن هذا القيد قد تمَّ التحقق منه بشكله الفوري.

```
SET CONSTRAINTS Emp_FK DEFERRED;
INSERT INTO Employee_T values(100,..,1);
INSERT INTO Department_T values(1,..,100);
COMMIT;
```

شكل رقم (7–5): كسر «القيود الحلقية» باستخدام طريقة تأخير العمل بالقيود.

7–1–1–4 إنشاء منظور (Create View):

تُمكِّن لغة الاستفسار البنائية من تعريف المنظورات. والمنظور عبارة عن جدول تخيُّلي (VIRTUAL TABLE) له تعريف ضِمن هيكل قاعدة البيانات؛ ولكنه لا يحتوي على بيانات مُخزَّنة فيه بشكلٍ دائم، وإنما يستمدُّ بياناته من الجداول (الأساسية) الموجودة في قاعدة البيانات. ويقوم نظام إدارة قاعدة البيانات بتعبئة البيانات المناسبة للمنظور بمجرد التعامل معه (من خلال إجراء عملية اختيار، أو إضافة، أو تحديث عليه) من أحد المستفيدين المخوَّلين بالتعامل معه. وبمجرد الانتهاء من التعامل مع المنظور؛ يقوم نظام إدارة قاعدة البيانات بإتلاف ما يحتويه من بيانات (بعد إجراء التعديلات المطلوبة، إن وُجِدت، على الجدول أو الجداول الأساسية الذي استمدَّ المنظور بياناته منها). وعند التعامل مع المنظور مرةً أخرى؛ تتمُّ تعبئته بالبيانات المناسبة مرةً أخرى، وهكذا. غير أنَّ بعض نظم قواعد البيانات تقوم بالمحافظة على بيانات المنظور لبعض الوقت عوضاً عن إتلافها بمجرد انتهاء مُستخدِمٍ ما من الانتهاء من التعامل مع المنظور؛ وذلك على أمل أن يأتي مستفيدٌ آخر يرغب في التعامل مع بيانات المنظور. وتُسمَّى الطريقة الثانية بالمنظورات المخزَّنة (Materialized Views). وتُعَدُّ هذه الطريقة مفيدةً في بعض الحالات؛ لأنها تعفي من تعبئة البيانات للمنظور كلما استدعت الحاجة الرجوع إليه، وخاصةً أن تكلفة تعبئة بيانات المنظور قد تكون كبيرةً جداً؛ من حيث الوقت اللازم من الحاسب الآلي لتعبئتها.

وتتمثل أهمية المنظورات في شكلين: الأول منهما عندما يكون هنالك عمليات اختيار معقدة قد تكون مصدراً للأخطاء عند محاولة كتابتها بشكلٍ متكررٍ من قِبل المستفيدين أو في برامج التطبيقات، والثاني منهما يتمثل في كون المنظورات توفر حمايةً للبيانات من الاستخدامات التي لا يُرْغَب في التصريح بها لبعض المستفيدين. فعلى سبيل المثال: يُمكِن كتابة منظور يحتوي على البيانات المالية لأعضاء هيئة التدريس في الجامعة الأهلية، ومنظور آخر لا يحتوي على مثل هذه البيانات؛ بحيث تُعطَى صلاحية التعامل مع المنظور الأول للمستفيدين في إدارة الشئون المالية

وبرامج التطبيقات المالية؛ في حين تُعطَى صلاحيات التعامل مع المنظور الثاني للمستفيدين في إدارة القبول والتسجيل ورؤساء الأقسام العلمية؛ سواء بشكل مباشر أو من خلال التطبيقات التي يستخدمونها للتعامل مع قاعدة البيانات. ولتعريف مثل هذين المنظورين يمكِن استخدام تعليمة الاختيار (التي سيتمُّ شرحها بالتفصيل في الجزء 7-2-1)، كما يلي:

```
CREATE VIEW FACULTY_FINANTIAL_INF_V AS
SELECT Faculty_ID, FName, LName, Salary, Department_ID
FROM FACULTY_T;
```

وعند استعراض محتويات المنظور السابق باستخدام تعليمة الاختيار على المنظور، كما يلي:

```
SELECT *
FROM FACULTY_FINANTIAL_INF_V;
```

تكون النتيجة جميع الحقول التي تهم المتعاملين مع المنظور في إدارة الشئون المالية، كما يلي:

FACULTY_	FNAME	LNAME	SALARY	DEPART
200	Khalid	Aloufi	35000	MATH
220	Fahad	Alhamid	25900	MATH
310	Saleh	Aleesa	30000	CS
320	Mohammed	Alhamad	44000	CS
330	Ghanim	Alghanim	44500	CS
340	Ibraheem	Alsaleh	25000	CS
400	Ahmad	Alotaibi	33900	CHEM
420	Saleh	Alghamdi	44600	CHEM
500	Yahya	Khorshid	36700	ENGL
540	Salem	Alhamad	40000	ENGL
560	Salman	Albassam	33800	ENGL
600	Turki	Alturki	27800	STAT
640	Fahad	Alzaid	44300	STAT
660	Saud	Alkhalifa	44900	STAT
710	Mahmood	Alsalem	31900	PHYS
730	Mishal	Almazid	29800	PHYS
770	Sultan	Aljasir	43300	PHYS
800	Ali	Albader	45300	EE
810	Saad	Alzhrani	44200	EE
850	Ahmad	Alsabti	33900	EE

أمَّا المنظور الثاني الذي لا يحتوي على البيانات المالية؛ فيمكن تعريفه، كما يلي:

```
CREATE VIEW FACULTY_V AS
SELECT Faculty_ID, FName, LName, DOB, Phone_NO, Department_ID
FROM FACULTY_T;
```

ويمكن استعراض محتويات المنظور السابق باستخدام تعليمة الاختيار على المنظور، كما يلي:

```
SELECT *
FROM FACULTY_V;
```

وتكون النتيجة جميع الحقول التي تهمُّ المتعاملين مع المنظور من غير العاملين في إدارة الشئون المالية، كما يلي:

FACULTY_	FNAME	LNAME	DOB	PHONE_NO	DEPART
200	Khalid	Aloufi	22-MAY-63	454-2341	MATH
220	Fahad	Alhamid	07-OCT-70	456-7733	MATH
310	Saleh	Aleesa	13-SEP-66	454-8932	CS
320	Mohammed	Alhamad	13-MAY-65	454-5412	CS
330	Ghanim	Alghanim	12-AUG-69	456-2234	CS
340	Ibraheem	Alsaleh	20-JAN-70	454-1234	CS
400	Ahmad	Alotaibi	17-MAY-71	454-4563	CHEM
420	Saleh	Alghamdi	13-FEB-69	454-2233	CHEM
500	Yahya	Khorshid	12-MAR-65	456-2221	ENGL
540	Salem	Alhamad	11-SEP-72	456-3304	ENGL
560	Salman	Albassam	13-SEP-68	454-7865	ENGL
600	Turki	Alturki	23-JUL-75	456-7891	STAT
640	Fahad	Alzaid	12-MAY-71	456-3322	STAT
660	Saud	Alkhalifa	13-AUG-72	454-9856	STAT
710	Mahmood	Alsalem	19-FEB-73	456-3323	PHYS
730	Mishal	Almazid	17-SEP-75	454-2343	PHYS
770	Sultan	Aljasir	13-MAY-70	456-3212	PHYS
800	Ali	Albader	22-JUN-66	456-7812	EE
810	Saad	Alzhrani	17-OCT-67	454-5578	EE
850	Ahmad	Alsabti	15-APR-73	456-0120	EE

ويجسِّد المنظور في لغة الاستفسار البنائية مفهوم عدم الاعتمادية المنطقية، الذي تمَّ التطرق إليه في الفصل الأول، بين المنظورات الخارجية (External Views)، والمنظور المفاهيمي (أو

المنطقي) لقاعدة البيانات؛ بحيث إن أيَّ تغيير للجداول المكوِّنة لقاعدة البيانات لا يؤثر في نظرة المستفيدين لقاعدة البيانات. فعلى سبيل المثال: عند تجزئة جدول إلى جدولين أو أكثر أو إضافة حقول جديدة للجدول؛ فإن المستفيدين، باستخدام المنظورات، قد لا يلحظون مثل هذا التغيير في هيكل قاعدة البيانات. لهذا السبب؛ فإن إداريِّ قواعد البيانات العلاقية كثيراً ما يُقرنون بين كلِّ جدول من جداول قاعدة البيانات بمنظور له. ويتمُّ استخدم المنظور المصاحب لجدول ما من خلال التعليمات نفسها التي تُستخدَم مع الجداول دون أيِّ تفريق كما لو أنه يتمُّ استخدام الجدول الأساسي، وليس المنظور المصاحب له. ويعني هذا أنه يتمُّ استخدام المنظور من قِبل المستفيدين ومن قِبل التطبيقات المبنية على قاعدة البيانات عوضاً عن التعامل مع الجدول الأساسي بشكلٍ مباشر. وبهذه الطريقة يمكِن إجراء أيِّ تعديلات قد تطلبها مراحل مستقبلية على جداول قاعدة البيانات دون الحاجة لإجراء تعديلات مصاحبة على نظم التطبيقات أو التعليمات التي تَعوَّد المستفيدون على تنفيذها على قاعدة البيانات. وباستخدام المنظورات يُكتفى بتعديل تعريف المنظور الذي جرى التعارف على استخدامه من قِبل المستفيدين وبرامج التطبيقات من خلال إزالته، ومن ثم إعادة تعريفه من جديد تحت المُسمَّى نفسه. وبهذه الطريقة؛ فإنه لا يُوجَد ما يدعو إلى التعديل على برامج التطبيقات؛ حتى تنعكس التعديلات التي أُجرِيَت على الجداول الأساسية لقاعدة البيانات على برامج التطبيقات. الأمر الذي يعني وجود استقلالية (أو عدم اعتمادية منطقية) بين برامج التطبيقات والتصميم المنطقي لقاعدة البيانات.

وتجدر الإشارة إلى أنه لا يقتصر تعريف المنظور بحيث يقترن بجدول واحد فقط من جداول قاعدة البيانات، وإنما يمكِن تعريفه؛ بحيث يقترن بأكثر من جدول واحد. كما أنه من الممكن أن تُستخدَم دوال التجميع (التي سنتطرق إليها لاحقاً) في تعريف بعض حقول المنظور. فعلى سبيل المثال: يمكِن تعريف المنظور التالي الذي يقترن بأعضاء هيئة التدريس الذين يعملون في قسم الحاسب الآلي (في جدول أعضاء هيئة التدريس) والمواد الدراسية المؤهل لتدريسها هؤلاء الأعضاء (في جدول المؤهلات التدريسية)، كما يلي:

```
CREATE VIEW CS_FACULTY_QUALIFICATION_V AS
SELECT FACULTY_T. Faculty_ID, FName, LName, Course_ID, Date_Qualified
FROM FACULTY_T, QUALIFICATION_T
WHERE FACULTY_T.Faculty_ID = QUALIFICATION_T.Faculty_ID AND
        FACULTY_T. Department_ID = 'CS';
```

تعرف التعليمة السابقة منظوراً باسم «المؤهلات التدريسية لأعضاء هيئة التدريس الذين يعملون في قسم الحاسب الآلي»، التي يُلاحَظ فيها استخدام الحرف "V" للدلالة على تعريف منظور عوضاً عن جدول؛ بحيث يقترن المنظور بجدولين كما أسلفنا أعلاه. وعند الاستفسار عن المحتويات التي يقترن بها هذا المنظور في الجداول الأساسية من خلال التعليمة التالية:

```
SELECT *
FROM CS_FACULTY_QUALIFICATION_V;
```

تكون نتيجة التعليمة السابقة التي تسترجع محتويات المنظور حسب تعريفه أعلاه، كما يلي:

```
FACULTY_ FNAME        LNAME        COURSE_ DATE_QUAL
-------- ------------ ------------ ------- ---------
310      Saleh        Aleesa       CS101   05-JUN-95
320      Mohammed     Alhamad      CS102   09-AUG-95
320      Mohammed     Alhamad      CS103   03-AUG-96
330      Ghanim       Alghanim     CS104   02-SEP-97
340      Ibraheem     Alsaleh      CS105   02-DEC-97
```

ولأنه من الممكِن تعريف المنظورات؛ بحيث تقترن بأكثر من جدول، كما يمكِن أن تحتوي تعاريفها على دوال تجميع؛ فإنه قد يتعذر إجراء عمليات التعديل (من حذف أو إضافة أو تحديث) على بياناتها بمعنى عدم إمكانية تعديل بيانات الجداول المقترنة بـالمنظورات. والقاعدة العامة التي تمكِّن من إجراء عمليات التعديل على بيانات المنظورات هي: يمكن إجراءُ عمليات التعديل على بيانات منظور ما دام للقِيم التي أُجْرِي التعديلُ عليها في المنظور ما يكافئها من حقول في الجداول المقترنة بتعريف المنظور دون أيِّ التباس بـين هذه الحقول. وبناءً على هذه القاعدة؛ يتعَذَّر بشكلٍ عام إجراء التعديل على أيِّ منظور يكون من ضِمن حقوله حقولٌ يُستخدَم في تعريفها «دوال تجميع» (Aggregation FunCtions). والسبب وراء ذلك؛ هو عدم اقتران الحقول الممثلة

لدوال التجميع، ضمن حقول المنظور، بأيٍّ من حقول الجداول التي يقترن بـها المنظور. وفي مثل هذه الحالة يمكِن إجراء عمليات الاختيار (أو الاسترجاع) فقط على مثل هذه المنظورات.

5–1–1–7 إنشاء فهرس (Create Index):

الفهرس، بشكلٍ عام، عبارة عن طريقة تمكِّن من الوصول إلى الشيء المطلوب بسرعة كبـيرة. وتُستخدَم الفهرسة بشكلٍ مُكثَّف في حياتنا اليومية؛ حيث نجدها مُستخدَمة، على سبيل المثال، في المكتبات، والمستوصفات (أو المستشفيات) الطبـية... إلخ. وبدون الفهارس يضطر الشخص الذي يبحث عن كتاب في مكتبةٍ ما، على سبيل المثال، إلى السير في ممرات المكتبة كافةً بشكلٍ متسلسل وقراءة عنوان كلِّ كتاب في كلِّ ممرٍّ وبشكلٍ متسلسل أيضاً حتى يجد الكتاب الذي يبـحث عنه. وفي حالة كان الشخصُ يبحث عن كتاب ليس موجوداً أصلاً في المكتبة؛ فإنه سيضطر إلى المرور بكافة الكتب الموجودة؛ حتى يتيقن من عدم وجود الكتاب. على النقيض من ذلك؛ فإن الفهارس تعجِّل في عملية البحث وبشكلٍ كبيرٍ عن الشيء المطلوب. وتقتضي الفهارس أمرين: (1) وجود تصنيف معين للأشياء الموجودة وترتيب الأشياء وفقاً للتصنيف، و(2) ترتيب الأشياء وفق نمطٍ معين على أرض الواقع. فعلى سبيل المثال: قد يتمُّ ترتيب الكتب في المكتبة وفق المقياس الدولي لترقيم الكتب (International Standard Book Number (ISBN))، ويُبنَى على هذا الترتيب فهارس مختلفة من ضِمنها فهرس الموضوعات، وفهرس المؤلفين، وفهرس دور النشر... إلخ. ومعنى هذا أننا قد قمنا بترتيب الكتب في المكتبة وفق نمط معين على أرض الواقع، وأنشأنا فهارس للتصنيفات المناسبة. في هذه الحالة يمكِن البحث عن أيِّ كتاب وفق أحد المعايير التي صُنِّفت عليها. فمثلاً؛ يمكِن البحث عن الكتب التي أُلِّفت من قِبل مؤلف ما في فهرس المؤلفين. وحيث إنَّ هذا الفهرس مُرتَّب أبجدياً حسب أسماء المؤلفين؛ فإن عملية البحث ستتم بشكلٍ أسرع؛ لأن عملية البحث لن تتم بالضرورة بـشكلٍ متسلسل في الفهرس. وعند العثور على الكتاب ضِمن فهرس المؤلفين؛ يتم التعرُّف على رقمه. بعد ذلك يتمُّ الذهاب للممرِّ الذي يحتوي على مكان الكتب الذي يتضمَّن رقم الكتاب المطلوب، والسير في ذلك الممرِّ بشكلٍ متسلسل حتى العثور على الكتاب. أمَّا في حالة البحث عن كتاب لمؤلفٍ ما والكتاب غير متوفر في المكتبة؛ فإنه يُكتفى بالبحث في فهرس المؤلفين. وعند عدم العثور على اسم المؤلف ضِمن أسماء المؤلفين، يتمُّ التوقف عن البحث لعدم توفر الكتاب في المكتبة.

والفهرس في نظم قواعد البيانات ما هو إلا هيكل بيانات (Data StruCture) يُبنَى على حقل أو أكثر من حقول الجدول تمثل فكرته الرئيسية ما نجده من فهارس في حياتنا اليومية. وتساعد الفهارس في الوصول إلى السجل المطلوب في جدولٍ ما خاصة عندما يتمُّ بناؤه على حقل معين وتكون عمليات الاستفسار تتضمَّن مقارنة بين الحقل الذي بني عليه الفهرس وقيمة ثابتة مثل «رقم السجل المدني = 123456789»؛ بحيث إن رقم السِّجل المدني هو الحقل الذي بنِي عليه الفهرس في جدول الموظفين، على سبيل المثال.

وعلى الرَّغم من أن تعليمة إنشاء الفهارس لم تَعُد من ضِمن لغة الاستفسار البنائية لكونها لا تتعلق بتعريف هياكل قاعدة بيانات أو تداول محتوياتها وإنما وسيلة لتسريع وتحسين أداء نظم إدارة قواعد البيانات؛ فإنَّ غالبية نظم إدارة قواعد البيانات العلاقية تحتوي على تعليمة لإنشاء الفهارس. وتستخدم عبارة إنشاء فهرس (Create Index) حسب الشكل العام التالي للتعليمة:

```
CREATE [UNIQUE] INDEX IndexName
ON TableName
(ColumnName [order][,ColumnName [order]] .... );
```

- تُستخدَم الكلمة الاختيارية فريد [UNIQUE] عند الرغبة في إنشاء فهرس يكون الحقل الذي أُنشِئ عليه الفهرس حقلاً فريداً لا تتكرر قِيمه في سجلات الجدول.

- يُمكِن تحديد أكثر من عمود لإنشاء فهرس مركب.

- يُقصَد بالترتيب [order] الطريقة التي سيتمُّ فيها ترتيب الفهرس؛ فإما أن يكون تصاعدياً (ASC) أو تنازلياً (DESC)؛ بحيث يكون الترتيب الافتراضي هو الترتيب التصاعدي.

- يُمكِن إنشاءُ أيِّ عددٍ من الفهارس للجدول الواحد؛ سواءٌ أكانت مبنيةً على حقول فريدة (UNIQUE) أو حقول يُسمَح فيها بالقيم المتكررة (DupliCates).

ولإنشاء فهرسٍ فريد باسم فهرس الطلبة (Student_IDX) على الرقم الدراسي للطلبة في جدول الطلبة بشكلٍ تصاعدي تُستخدَم تعليمة إنشاء فهرس، كما يلي:

```
CREATE UNIQUE INDEX STUDENT_IDX
ON STUDENT_T (Student_ID);
```

أمَّا إذا أردنا إنشاء فهرس بأسماء الطلبة تحت مُسمًّى (Student_Names_IDX) على اسم العائلة للطلبة بشكلٍ تصاعدي، والاسم الأول لهم بشكلٍ تنازلي فتُستخدَم التعليمة التالية:

```
CREATE INDEX STUDENT_NAMES_IDX
ON STUDENT_T (LName ASC, FName DESC);
```

2–1–7 تعليمة الإزالة (DROP STATEMENT):

تُستخدَم تعليمة «الإزالة» (DROP) لحذف العناصر ذات المُسمَّيات من هيكل قاعدة البيانات مثل الجداول، والمجال، والقيود، والمنظورات، والفهارس... إلخ. كما أنه يُوجَد نوعان من الخيارات السُّلوكية لتعليمة الإزالة وهي «التتابع» (CASCADE)، و«التقييد» (RESTRICT). فإذا ما أردنا حذف (تعريف) قاعدة بيانات بأكملها بما في ذلك ما تحتويه من جداول وقيود وجميع العناصر الأخرى المكوِّنة لقاعدة البيانات يمكِن استخدام تعليمة الإزالة مصحوبة بخيار التتابع، كما يلي:

```
DROP SCHEMA Company CASCADE;
```

أمَّا إذا تمَّ استخدامُ خيار التقييد (RestriCt)، الذي يمثل الحالة الافتراضية، ضمن تعليمة الإزالة؛ فإنه يتمُّ إزالة (تعريف) قاعدة البيانات فقط، وذلك عند عدم وجود أيِّ عنصر فيها. أما إذا وُجِد فيها عنصرٌ أو أكثر فلن تنفذ عملية الإزالة.

كذلك هو الحال عند استخدام التعليمة لإزالة جدولٍ ما؛ حيث يتمُّ استخدام خيار التتابع لإزالة محتويات الجدول، وتعريفه، والقيود المفروضة عليه، والمنظورات المبنية عليه. أمَّا إذا تمَّ استخدامُ خيار التقييد؛ فإن عملية الإزالة لا تتم إلا في حالة عدم وجود ما يرتبط بالجدول من عناصر أخرى ضِمن قاعدة البيانات. كما يُمكِن استخدام تعليمة الإزالة لحذف أيٍّ من مكوِّنات

قاعدة البيانات الأخرى، مثل: المنظورات، والقيود، والفهارس، وقيود المجال التي يتمُّ تعريفها. وكما أسلفنا أعلاه؛ فإن الحالة الافتراضية لعملية الإزالة هي التقييد، بمعنى عدم حذف العنصر إذا احتوى على أية بيانات أو ارتبط بأيٍّ من العناصر الأخرى لقاعدة البيانات. ويمثل الشكل التالي ثلاث تعليمات: الأولى تمثل عملية إزالة جدول المواد الدراسية (Course_T)، على افتراض عدم وجود أيِّ سجلات في الجدول، والثانية لإزالة فهرس بمُسمَّى (Course_IDX)، والثالثة لإزالة منظور بمُسمَّى (Course_V).

```
DROP TABLE COURSE_T;
DROP INDEX COURSE_IDX;
DROP VIEW COURSE_V;
```

3-1-7 تعليمة التعديل (ALTER STATEMENT):

إنَّ تعريف أيِّ جدول أو عنصر ذي مُسمًّى ضِمن قاعدة البيانات يمكِن التعديل عليه باستخدام تعليمة التعديل (ALTER). ومن التعديلات التي بالإمكان إجراؤها على جدول ما إضافة حقل جديد، أو حذف حقل من حقول الجدول، أو تغيير تعريف حقل ما ضِمن الجدول، أو إضافة أو حذف قيد. فعلى سبيل المثال: يمكِن إضافة حقل جديد لجدول أعضاء هيئة التدريس يعكس تاريخ الحصول على آخر درجة علمية تكون بياناته من نوع تاريخ، كما يلي:

```
ALTER TABLE FACULTY_T ADD Graduation_Date Date;
```

وبعد تعريف الحقل الجديد؛ يجب إدخال بياناته إمَّا من خلال استخدام عبارة القيمة الافتراضية (DEFAULT)، أو من خلال استخدام تعليمة التحديث (UPDATE) على البيانات التي سنتطرق إليها لاحقاً. أما إذا لم يتم تعريف قيمة افتراضية للحقل الجديد؛ فستكون قيمته لجميع السجلات في الجدول غائبة.

ولإزالة حقلٍ ما من جدول؛ فإنه يجب تعريف سُلوك عملية الإزالة: إما تتابع (CASCADE) وإما تقييد (RESTRICT). وعند استخدام تتابع؛ تتمُّ إزالة الحقل والقيود

المعرفة عليه كافة، والمنظورات التي تستخدمه في تعريفها؛ وذلك بـشكلٍ تلقائي في أثناء إزالة الحقل. أمَّا إذا تمَّ استخدام خيار التقييد؛ فتكون عملية الإزالة ناجحة فقط عندما لا يكون هنالك أيٌّ عنصر من عناصر قاعدة البـيانات مستخدماً لهذا الحقل في تعريفه أو الرجوع إليه. وتمثل التعليمة التالية طريقة إزالة حقل تاريخ الميلاد (DOB) من جدول أعضاء هيئة التدريس:

```
ALTER TABLE FACULTY_T DROP DOB;
```

كما أنَّ تعليمة التعديل تمكِّن أيضاً من حذف القيمة الافتراضية لحقلٍ ما، كما يوضح المثال التالي الذي يقوم بإزالة القيمة الافتراضية من حقل رقم عضو التدريس المعرَّف في جدول المجموعات الدراسية:

```
ALTER TABLE SECTION_T ALTER FACULTY_ID DROP DEFAULT;
```

أمَّا تعليمة التعديل التالية؛ فتوضِّح طريقة تعريف قيمة افتراضية لحقل رقم عضو هيئة التدريس في جدول المجموعات الدراسية؛ بحيث تكون القيمة الافتراضية "AAAAAAAA":

```
ALTER TABLE SECTION_T ALTER FACULTY_ID SET DEFAULT 'AAAAAAAA';
```

ويُمكِن استخدامُ تعليمة التعديل؛ لتغيير القيود؛ من حيث حذفها أو تعريفها، كما سبق أن أوضحنا في الجزء المتعلق بـتوصيف القيود وأنواعها.

ويحتوي الملحق رقم (1) في جزئه السادس (ملحق رقم (1) - 6) على بناءٍ لجميع جداول قاعدة بيانات الجامعة الأهلية والبيانات التي تحتويها القاعدة في بـيئة أوراكل. وتمثل قاعدة البيانات هذه محور الغالبية العظمى من التمارين التطبـيقية الواردة في هذا الفصل وفي الفصل الثامن.

2-7 لغة معالجة البيانات (Data Manipulation Language (DML)):

7-2-1 تعليمة الاختيار (أو الاسترجاع) (SELECT STATEMENT):

تُعَدُّ تعليمة الاختيار (SELECT) واحدةً من أهمِّ تعليمات لغة الاستفسار البنائية. وتُستخدَم تعليمة الاختيار لاسترجاع البيانات الموجودة في قاعدة البيانات. كما تجدر ملاحظة أن تعليمة الاختيار في لغة الاستفسار البنائية ليست ذات أيَّة علاقة مع عملية الاختيار في الجبر العلاقي الذي سبق التطرُّق إليه في الفصل الرابع. كما تجدر أيضاً ملاحظة وجود فرق جوهري بين لغة الاستفسار البنائية والنموذج العلاقي الرسمي. ويَكْمُن هذا الفرق في أن لغة الاستفسار البنائية تتعامل مع الجداول على أساس أنها حقائب (Bags) (أو مجموعات متكررة (Multiset)) عوضاً عن مجموعات من السجلات كما هو الحال في النموذج العلاقي الرسمي. ويعني ذلك؛ أنه من الممكن أن تتكرَّر السجلات في الجداول باستخدام لغة الاستفسار البنائية؛ الأمر الذي لا يُمْكِن في النموذج العلاقي الرسمي. ومع هذا يمكِن أن تقيَّد الجداول؛ بحيث لا يمكن أن تتكرر السجلات فيها؛ وذلك باستخدام المفتاح الرئيسي الذي لا يمكن أن تتكرر قِيمه، كما يمكن أن تقيد القيم المسترجعة من جدول ما باستخدام تعليمة الاختيار؛ بحيث لا تتكرَّر ضِمن نتيجة العملية؛ وذلك باستخدام عبارة (DISTINCT) التي تقوم بحذف السِّجلات المتكرِّرة من نتيجة العملية.

ويُوجَد لتعليمة الاختيار العديدُ من الأشكال التي قد يكون بعضها مُعقداً بشكلٍ كبير؛ إلا أننا سنبدأ بأبسط أشكال التعليمة، وسنواصل شرح التعليمة وصولاً إلى الأشكال المعقدة منها. أمَّا الشكل العام للتعليمة فهو، كما يلي:

```
SELECT [DISTINCT] ColumnName(s)
FROM Table(s)
[WHERE Condition]
[GROUP BY ColumnName(s)]
[HAVING Condition]
[ORDER BY ColumnName(s)];
```

وتدلُّ العبارات داخل الأقواس المربعة، في الشكل العام للتعليمة، على أنها عبارات اختيارية يتمُّ استخدامها حسب الحاجة؛ مما يعني أنه لا يجب استخدامها في أشكال التعليمة كافة. إلا أنه عند استخدام العبارات الاختيارية؛ فإنه يجب إدراجها ضِمن التعليمة بالترتيب نفسه الوارد في الشكل العام للتعليمة المذكور أعلاه.

7-2-1-1 اختيار أعمدة مُحدَّدة من جدول:

إن أبسط شكلٍ لتعليمة الاختيار (SELECT)؛ يكون عند استخدامها لاختيار حقل أو أكثر من جدول معين. وعند استخدام التعليمة في هذه الصورة؛ فإنه يعني إظهار قيم الحقول التي تمَّ تحديدها ضِمن عبارة الاختيار (SELECT) من الجدول المذكور في عبارة «من» (FROM). ويبيّن الشكل التالي تعليمة الاختيار في أبسط صورها.

```
SELECT TableName.ColumnName,
TableName.ColumnName, ...
;FROM TableName
```

- مثـــال 1: ما أرقام وأسماء المواد الدراسية التي تقدِّمها الجامعة الأهلية؟

الحل: نستخدم تعليمة الاختيار (SELECT) على جدول المواد الدراسية (COURSE_T)؛ بحيث يتمُّ اختيار حقل رمز المادة الدراسية (Course_ID) وحقل عنوان المادة الدراسية (Title)، كما يلي:

```
SELECT COURSE_T.Couse_Id,
COURSE_T.Title
FROM COURSE_T;
```

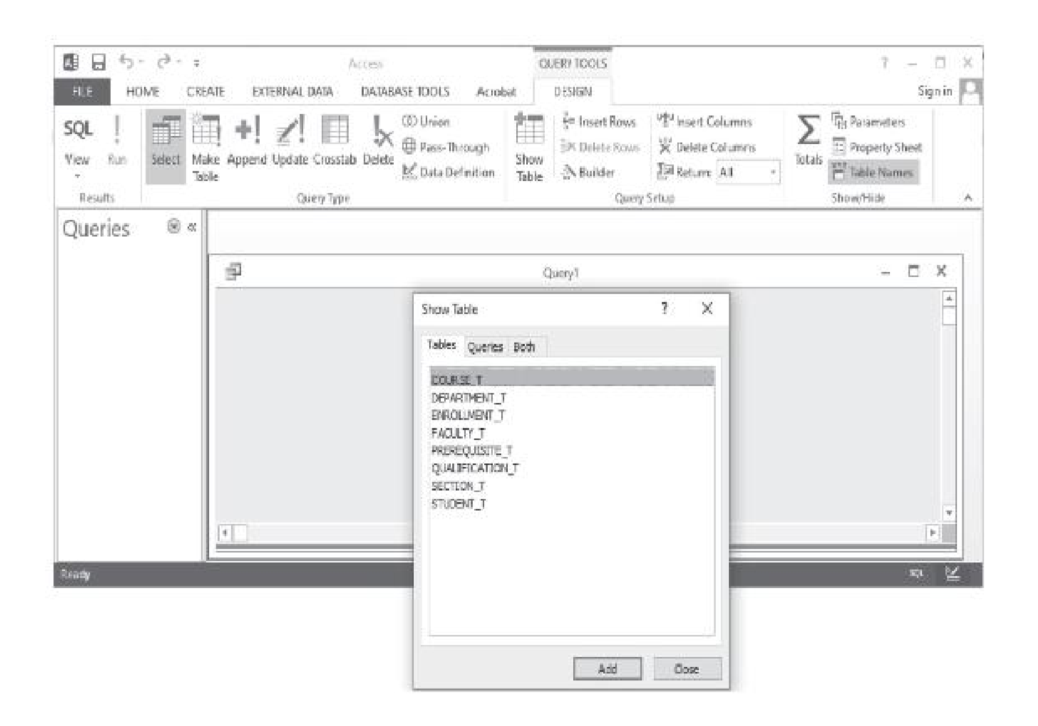
ويُمكِن الاستغناء عن اسم الجدول في تعليمة الاختيار (SELECT) (أو غيرها من تعليمات (SQL)) من أسماء الحقول؛ وذلك في حالة عدم وجود التباس في أسماء الحقول التابعة للجداول المُستخدَمة في التعليمة. ففي المثال السابق جميع الحقول تتبع لجدول واحد، هو جدول المواد الدراسية (COURSE_T)؛ لذلك فإنه لا يُوجَد التباس في مسميات الحقول التي سيتم تطبيق تعليمة الاختيار عليها؛ لذا فإنه يمكن الاستغناء عن اسم الجدول من مسميات الحقول المختارة لتصبح التعليمة كالتالي:

```
SELECT Course_Id, Title
FROM COURSE_T;
```

وباستخدام نظام قاعدة بيانات أكسس يمكِن تنفيذ التعليمة من خلال اختيار أيقونة تصميم الاستعلامات (Query Design) الموجودة تحت تبويبة الإنشاء (Create)، كما يلي:



وعند الضغط على الأيقونة؛ تظهر قائمة الجداول التي من الممكن أن يُنفَّذ عليها تصميم الاستفسار، كما يلي:

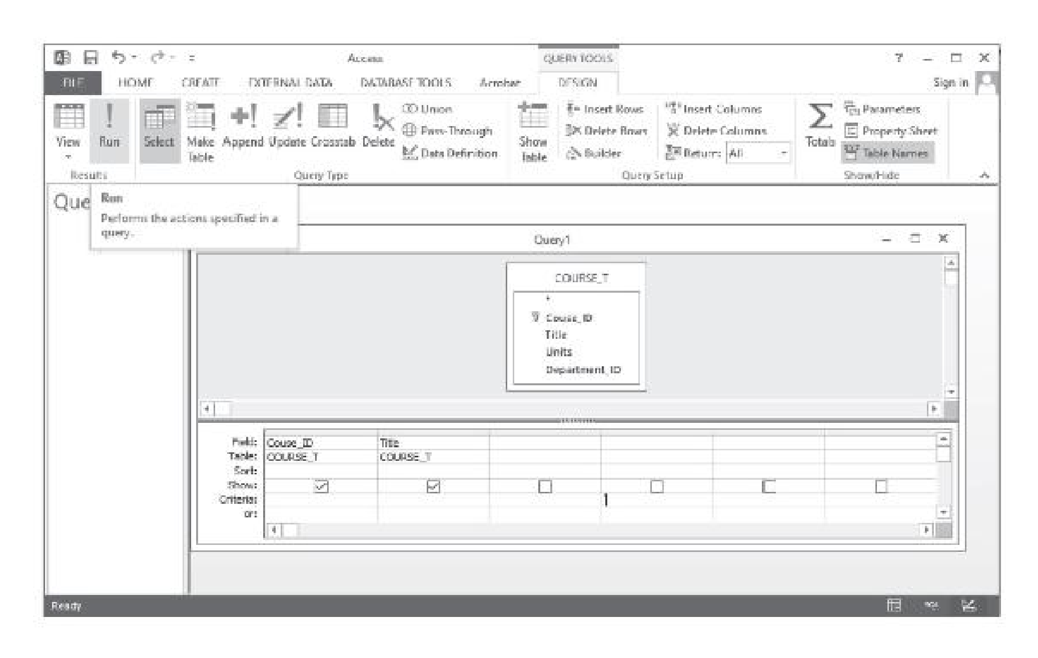


يتمُّ اختيار جدول المواد الدراسية، الذي ستُنفَّذ عليه تعليمة الاختيار، ويتمُّ الضغط على أيقونة الإضافة (Add). عندئذٍ سيتمُّ إدراج الجدول في الجزء العلوي من شاشة الاستفسار. بعد ذلك يتمُّ اختيار الحقلين المطلوبين؛ وذلك من خلال سحبهما من الجدول وإلقائهما في الحقل المناسب ضِمن المخطط السُّفلي المخصَّص لإنشاء التعليمة، أو الضغط على كلٍّ منهما باستخدام الفأرة ضغطاً مزدوجاً، كما يلي:

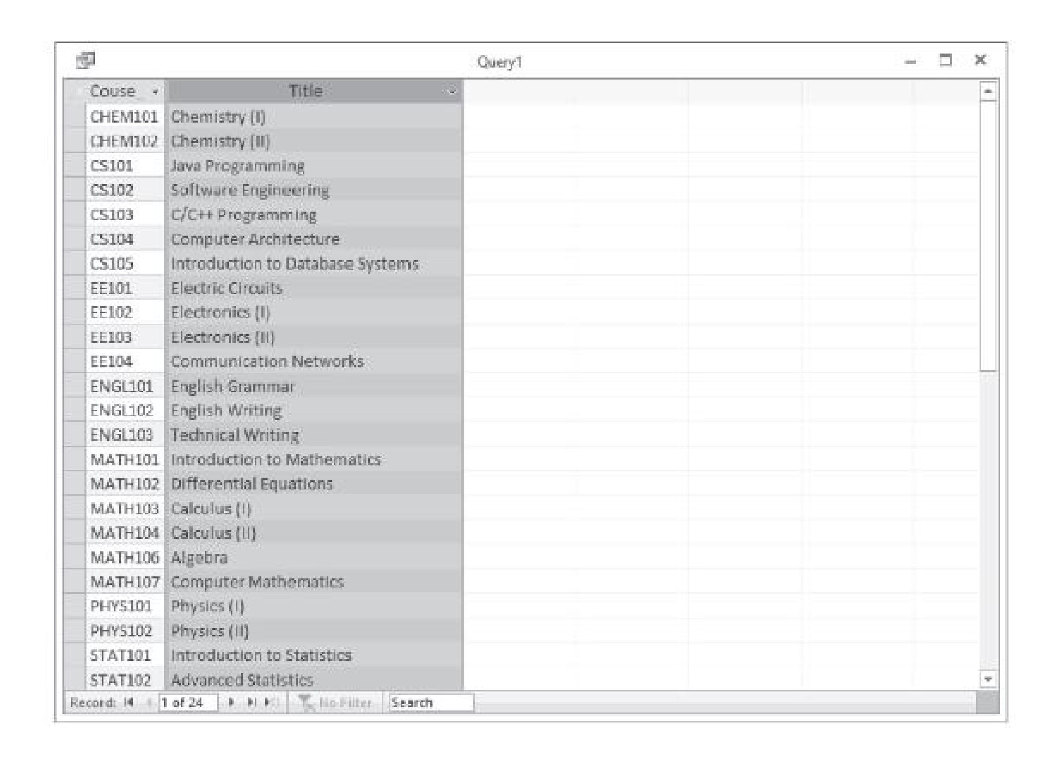


تجدرُ ملاحظة أن الحقل المُسمَّى رمز المادة الدراسية (Course_ID) قد ظهر وبجواره علامة مفتاح؛ للدلالة على أنه المفتاح الرئيسي للجدول. كما ظهر حقلٌ جديدٌ ضِمن الجدول برمز علامة النجمة (*). ويعني هذا الحقل أنه ممثلٌ لجميع حقول الجدول، بمعنى أنه يُمكِن اختياره لإظهار كل حقول الجدول ضِمن نتيجة عملية الاختيار دون الحاجة إلى سحب كلِّ حقل على حِدَة وإلقائه ضِمن مخطط إنشاء الاستفسار.

بعد ذلك يتمُّ تنفيذ عملية الاختيار من خلال اختيار أمر التنفيذ (RUN) الذي تعلوه علامة التعجب (!) من القائمة، كما يلي:

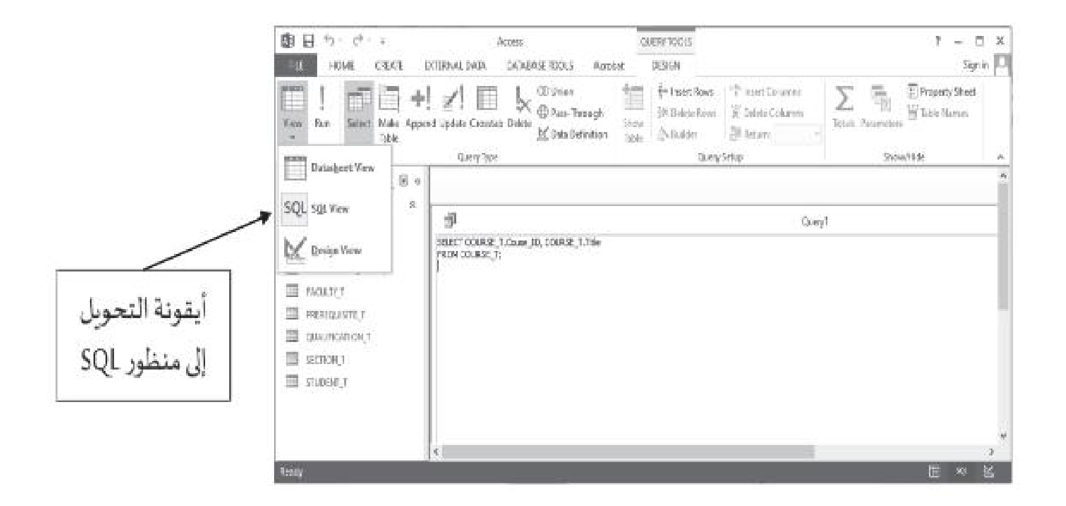


وبعد تنفيذ العملية تظهر نتيجة عملية الاختيار، كما يلي:



Couse	Title
CHEM101	Chemistry (I)
CHEM102	Chemistry (II)
CS101	Java Programming
CS102	Software Engineering
CS103	C/C++ Programming
CS104	Computer Architecture
CS105	Introduction to Database Systems
EE101	Electric Circuits
EE102	Electronics (I)
EE103	Electronics (II)
EE104	Communication Networks
ENGL101	English Grammar
ENGL102	English Writing
ENGL103	Technical Writing
MATH101	Introduction to Mathematics
MATH102	Differential Equations
MATH103	Calculus (I)
MATH104	Calculus (II)
MATH106	Algebra
MATH107	Computer Mathematics
PHYS101	Physics (I)
PHYS102	Physics (II)
STAT101	Introduction to Statistics
STAT102	Advanced Statistics

وتُمكِّن قاعدة بيانات أكسس أيضاً من إطلاع المستخدِم على تعليمة لغة الاستفسار البنائية المكافئة للتصميم الذي تمَّ إعداده من خلال الانتقال إلى شاشة عرض (SQL). كما يمكِن كتابة تعليمات (SQL) مباشرة من خلال شاشة عرض (SQL). والشكل التالي يُوضِّح تعليمة (SQL) لحلِّ المثال حسب ظهورها في شاشة عرض (SQL) التابعة لقاعدة بيانات أكسس بعد عملية تصميم الاستفسار.

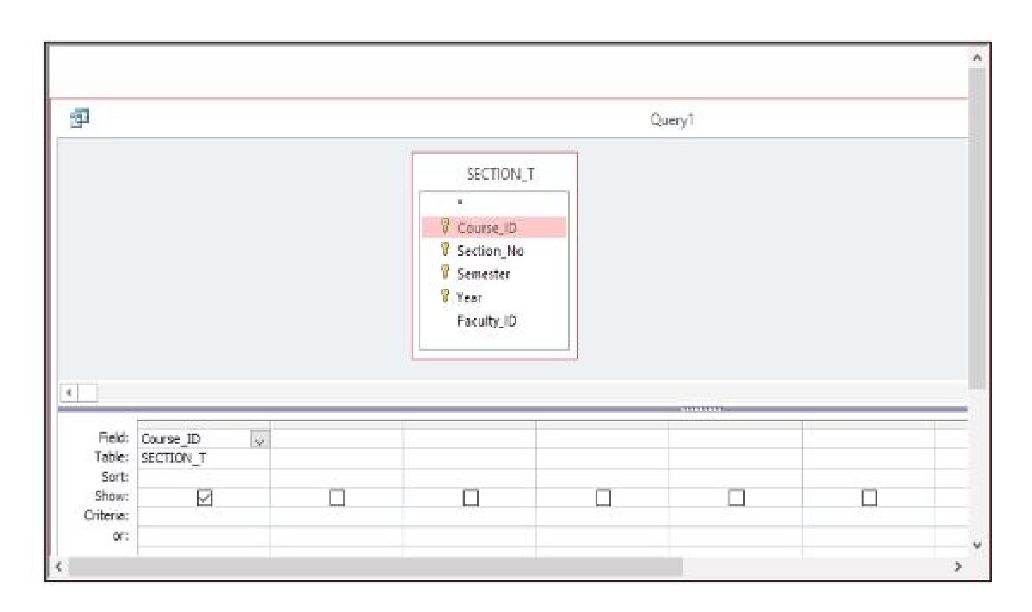


- مثـــال 2: ما أرقام المواد الدراسية المنفَّذة؟

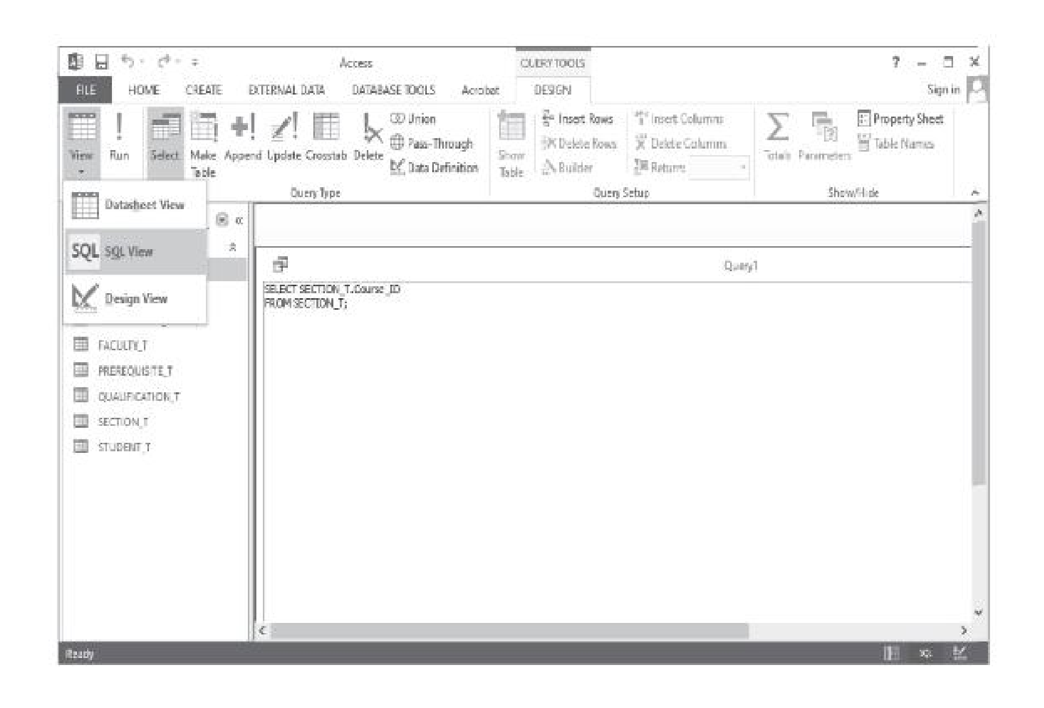
الحل: نستخدم تعليمة الاختيار (SELECT) مرةً أخرى؛ ولكن على جدول المجموعات الدراسية (SECTION_T) عوضاً عن جدول المواد الدراسية (COURSE_T)، وبحيث يتم اختيار حقل رمز المادة الدراسية (Course_ID)، كما يلي:

```
SELECT Course_ID
FROM SECTION_T;
```

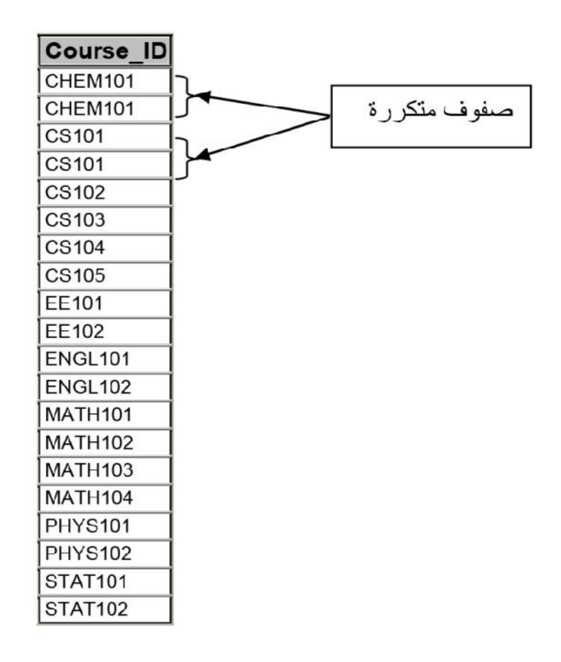
والشكل التالي يوضِّح شاشة عرض التصميم للتعليمة السابقة باستخدام نظام قاعدة بـيانات أكسس.



وللاطلاع على شكلِ التعليمة من منظور (SQL)؛ يتمُّ الانتقال إلى شاشة عرض (SQL)، كما في المثال السابق؛ ليظهر الشكل التالي للتعليمة.



وعند تنفيذ التعليمة باستخدام الأمر (RUN)؛ سواءٌ أكان ذلك من شاشة عرض التصميم أو من شاشة عرض (SQL)، يكون الناتج كما هو مُوضَّح في الجدول التالي.



Course_ID
CHEM101
CHEM101
CS101
CS101
CS102
CS103
CS104
CS105
EE101
EE102
ENGL101
ENGL102
MATH101
MATH102
MATH103
MATH104
PHYS101
PHYS102
STAT101
STAT102

يُلاحَظ في نتيجة التعليمة تكرار مادة الكيمياء (101) (CHEM101)، ومادة الحاسب الآلي (101) (CS101)؛ وذلك لكون كلتا المادتين تُنفذان من خلال مجموعتين دراسيتين عوضاً عن

مجموعة دراسية واحدة. ويعني هذا أن تعليمة الاختيار (SELECT)؛ تقوم باختيار قيمة الحقل الذي تمَّ تحديد اسمه ضِمن التعليمة دون اعتبار لافتراض ما إذا كانت قيمته قد تم اختيارها سابقاً ضِمن نتائج التعليمة أم لا؛ غير أنه في الكثير من الأحيان نحتاج إلى إدراج قِيم الصفوف في النتائج دون تكرار. وتظهر هذه الحاجة بشكلٍ خاص عندما لا يكون ضِمن الحقول المُحدَّدة في تعليمة الاختيار المفتاح الرئيسي للجدول؛ نظراً لإمكانية تكرار قيم بقية حقول الجدول، في مثل هذه الحالة، كما يُوضِّح المثال السابق. ولهذا السبب توفر مواصفات لغة الاستفسار البنائية (SQL) الكلمة المحجوزة (DISTINCT) التي تقوم بحذف الصفوف المتكرِّرة من النتيجة النهائية للتعليمة، وإظهار قيمة كلِّ صف مرة واحدة فقط بغض النظر عن المرات التي يتكرَّر فيها.

7-2-1-2 حذف الصفوف المتكرِّرة من نتيجة تعليمة الاختيار باستخدام كلمة (DISTINCT):

توفر (SQL) الكلمة المحجوزة (DISTINCT) ضِمن تعليمة الاختيار (SELECT). وتُمكِّن هذه الكلمة عند استخدامها من حذف تكرارات كلِّ صف وإظهاره ضمن النتيجة النهائية لتعليمة الاختيار مرةً واحدةً فقط بغض النظر عن عدد مرات تكرار الصف. وعند استخدام تعليمة الاختيار (SELECT) دون استخدام الكلمة (DISTINCT)؛ فإن القيمة الافتراضية للتعليمة هي إظهار صفوف النتيجة كافة بما فيها المتكرر منها. والشكل التالي يوضِّح تعليمة الاختيار عند تضمينها على كلمة (DISTINCT).

```
SELECT DISTINCT ColumnName, ColumnName, . . .
FROM TableName;
```

- مثـــال 3: ما أرقام المواد الدراسية المنفَّذة؟ أظهِر أرقام المواد الدراسية دون تكرار (أيْ: بغض النظر عن عدد المجموعات (أو الشُّعَب) المنفذة من خلالها).

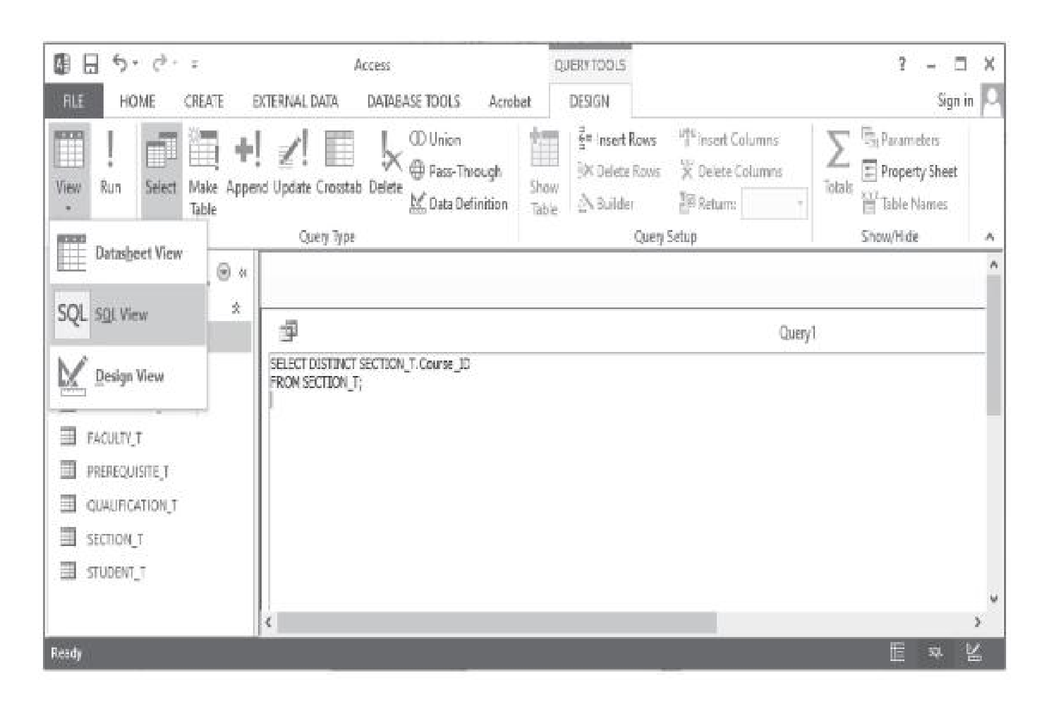
الحل: نستخدم تعليمة الاختيار (SELECT) مرةً أخرى على جدول المجموعات الدراسية (SECTION_T)؛ بحيث يتم اختيار حقل رمز المادة الدراسية (Course_ID) مع حذف الصفوف المتكررة من نتيجة الاختيار، كما يلي:

```
SELECT DISTINCT Course_ID
FROM SECTION_T;
```

وتكون نتيجة التعليمة السابقة، كما يلي:

```
COURSE_
-------
CHEM101
CS101
CS102
CS103
CS104
CS105
EE101
EE102
ENGL101
ENGL102
MATH101
MATH102
MATH103
MATH104
PHYS101
PHYS102
STAT101
STAT102
```

ويمكن تنفيذ التعليمة، في بيئة أكسس، مباشرة من خلال شاشة عرض (SQL)، كما يلي:



وتكون نتيجة التعليمة الجدول التالي الذي يُلاحَظ فيه حذف الصفوف المتكررة من نتيجة المثال السابق.

Course_ID
CHEM101
CS101
CS102
CS103
CS104
CS105
EE101
EE102
ENGL101
ENGL102
MATH101
MATH102
MATH103
MATH104
PHYS101
PHYS102
STAT101
STAT102

كما يُمكِن حلُّ المثال من خلال استخدام شاشة تصميم الاستفسار؛ بحيث يتمُّ تغيير خواص الاستفسار (Properties) بعد تصميم الاستفسار ليُظهِر القيم بشكلٍ غير متكرر، حسب التالي:

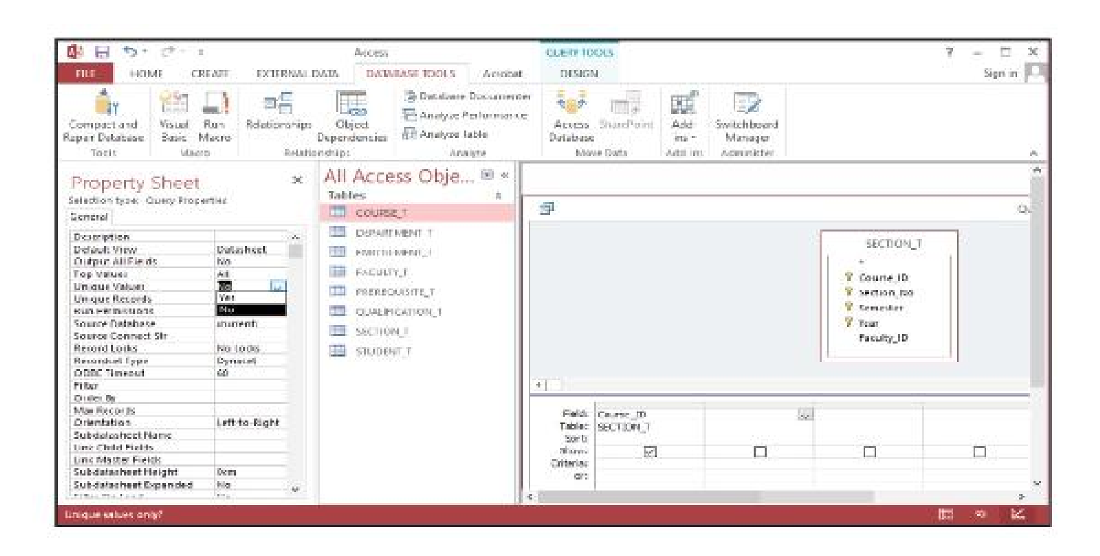
1- تصميم الاستفسار؛ بحيث يظهر رمز البرنامج (Course_ID) من جدول (SECTION_T).

2- الدخول إلى خصائص الاستفسار (Properties)؛ إمَّا من خلال القائمة أو بالضغط على زر الفأرة الأيمن (وذلك عندما يكون المؤشر في الجزء العلوى من شاشة التصميم) ومن ثم اختيار (Properties).

3- تغيير قيمة حقل القيم المتكررة (Unique Values) من (No) إلى (Yes).

4- تنفيذ الاستفسار.

ويوضِّح الشكل التالي شاشة عرض التصميم وقائمة الخواص مُوضّحاً عليها الحقل الواجب تغيير قيمته.



ويُمكِن التأكُّد من أن تصميم التعليمة في شاشة عرض التصميم مطابقٌ لتعليمة (SQL) المتضمِّنة على كلمة (DISTINCT)؛ من خلال الانتقال إلى شاشة عرض (SQL). كما أنه عند

تنفيذ التعليمة من شاشة عرض التصميم تكون القيم المُدرَجة في جدول النتيجة مطابقةً لنتائج تنفيذ تعليمة (SQL) من شاشة عرض (SQL).

7-2-1-3 الأسماء المُستعارة للأعمدة (ALIAS):

قد تحتوي أسماء الأعمدة في الجداول على اختصارات؛ بحيث يصعب على جميع المستخدِمين لقاعدة البيانات فهمها، أو قد يكون من المتعَارف عليه استخدامُ اسم معين من قِبل مجموعة من المستخدِمين لحقل معين يختلف عن الاسم الذي تمَّ تعريفه به في قاعدة البيانات. لذلك توفر لغة (SQL) القياسية طريقةً تمكِّن المستخدِم من إعادة تسمية أعمدة الجداول عند عرض نتائج تعليمة الاختيار (SELECT)؛ بحيث تظهر بمسميات مختلفة عن المسمَّيات الأصلية التي سُمِّيت بها في قاعدة البيانات. كما أن إعادة التسمية هذه لا تؤثر في المسميات الأصلية للأعمدة، وإنما تتلاشى فاعليتها بمجرد إظهار نتائج التعليمة. ولإعادة تسمية عمودٍ ما بكلمة واحدة تتمُّ كتابة الاسم الجديد له مباشرةً بعد اسمه الأصلي. أمَّا إذا كان الاسمُ الجديدُ مكوناً من أكثر من كلمة؛ فإن الاسم الجديد يُكتَب داخل علامتي تنصيص مزدوجة. والمثال التالي يُوضِّح طريقة استخدام إعادة التسمية.

- مثــال 4: ما أرقام وأسماء المواد الدراسية التي تقدِّمها الجامعة الأهلية؟ اجعل اسم عمود أرقام المواد الدراسية بمُسمَّى (CRS#) عوضاً عن مُسمَّاه الأصلي (Course_ID)، واسم عمود المواد الدراسية (Course Title) عوضاً عن مسماه الأصلي (Title).

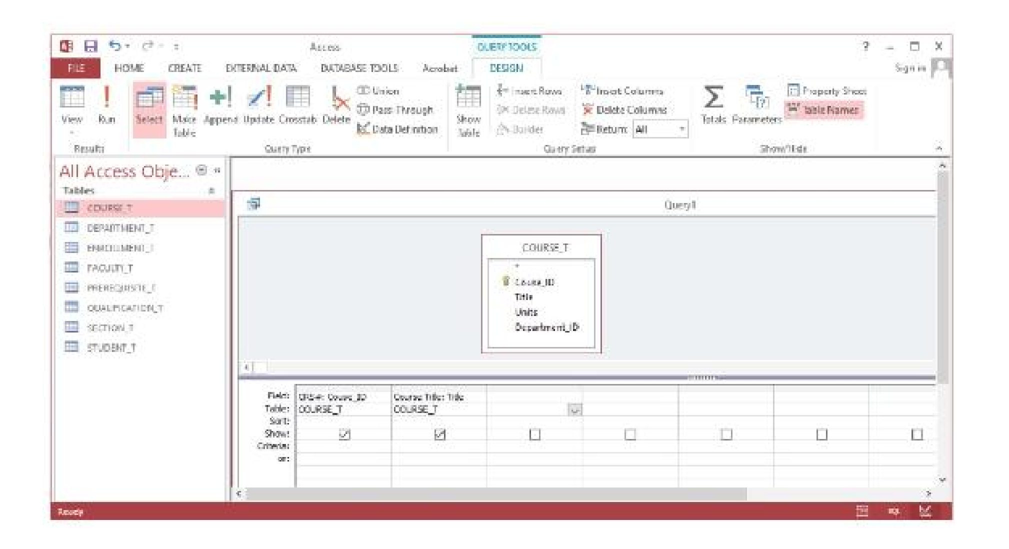
الحل: نستخدم تعليمة الاختيار (SELECT) على جدول المواد الدراسية (COURSE_T)؛ بحيث يتمُّ اختيار حقل رمز المادة الدراسية (Course_ID)، وحقل عنوان المادة الدراسية (Title). كما نستخدم طريقة إعادة التسمية المذكورة أعلاه، كما يلي:

```
SELECT Course_Id AS CRS#, Title AS “Course Title”
FROM COURSE_T;
```

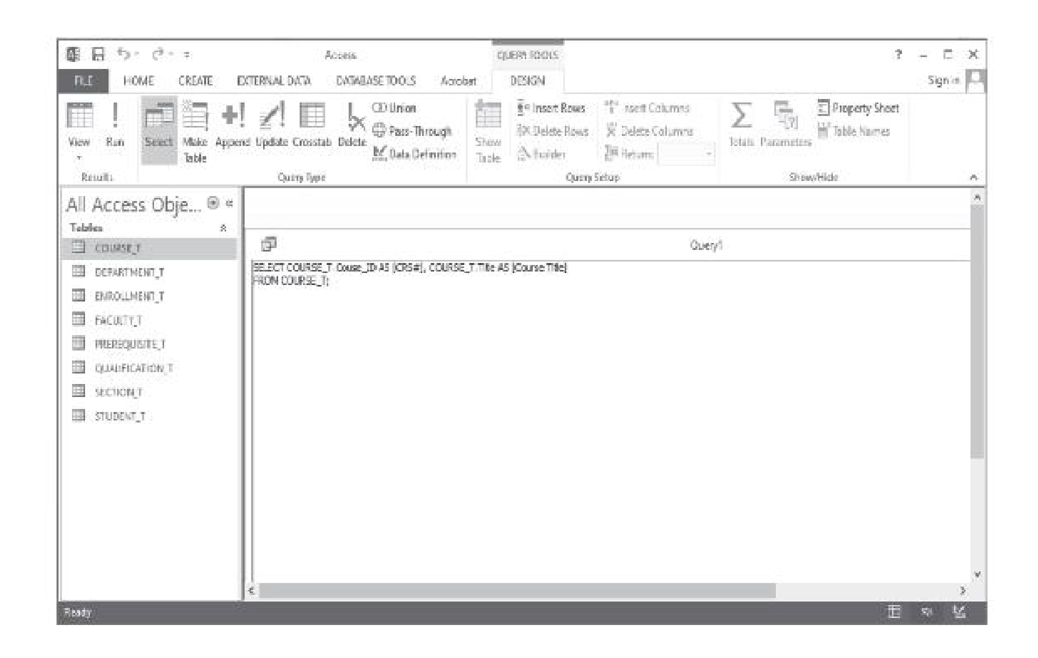
وتكون نتيجة التعليمة السابقة كما يلي:

```
CRS#    Course Title
------- -----------------------------------
CHEM101 CHEMISTRY (I)
CHEM102 CHEMISTRY (II)
CS101   JAVA PROGRAMMING
CS102   SOFTWARE ENGINEERING
CS103   C/C++ PROGRAMMING
CS104   COMPUTER ARCHITECTURE
CS105   INTRODUCTION TO DATABASE SYSTEMS
EE101   ELECTRIC CIRCUITS
EE102   ELECTRONICS (I)
EE103   ELECTRONICS (II)
EE104   COMMUNICATION NETWORKS
ENGL101 ENGLISH GRAMMAR
ENGL102 ENGLISH WRITING
ENGL103 TECHNICAL WRITING
MATH101 INTRODUCTION To MATHEMATICS
MATH102 DIFFERENTIAL EQUATIONS
MATH103 CALCULUS (I)
MATH104 CALCULUS (II)
MATH106 ALGEBRA
MATH107 COMPUTER MATHEMATICS
PHYS101 PHYSICS (I)
PHYS102 PHYSICS (II)
STAT101 INTRODUCTION TO STATISTICS
STAT102 ADVANCED STATISTICS
```

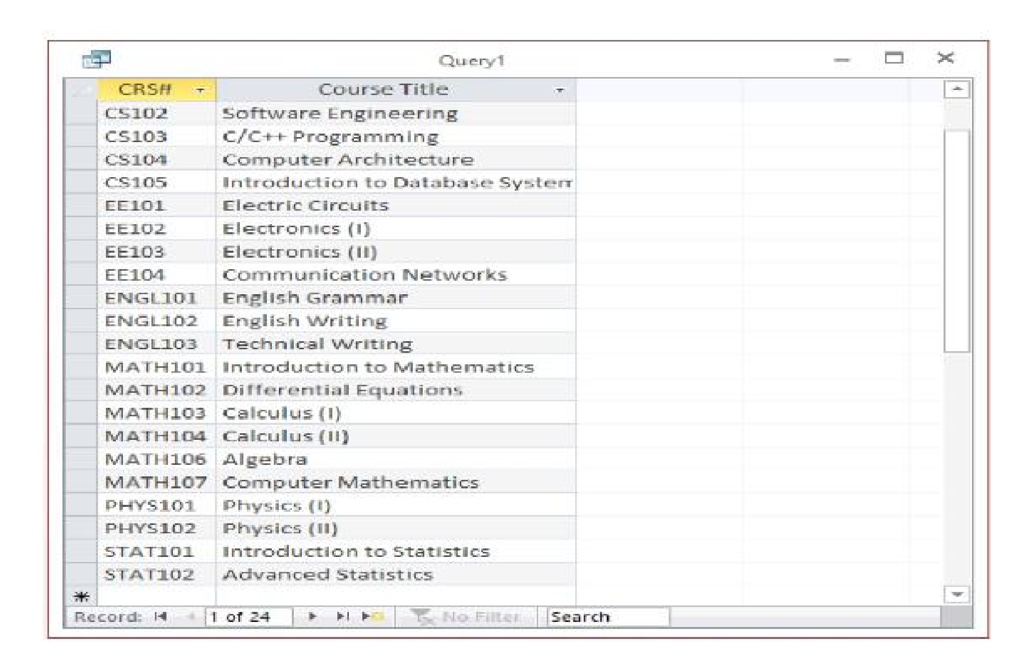
ويُمكِن تنفيذ التعليمة، في نظام أكسس؛ من خلال كتابة المُسمَّى الجديد للحقل متبوعاً بالنقطتين المتعامدتين (:) ومن ثم الاسم الأصلي للحقل، كما يلي:



وتظهر تعليمة لغة الاستفسار البنائية المُصمَّمة من خلال شاشة عرض التصميم في شاشة عرض (SQL)، كما يلي:



أما نتيجة تنفيذ التعليمة التي يُلاحَظ فيها تغيُّر مُسمَّيات الحقول، فتكون كما يلي:



CRS#	Course Title
CS102	Software Engineering
CS103	C/C++ Programming
CS104	Computer Architecture
CS105	Introduction to Database System
EE101	Electric Circuits
EE102	Electronics (I)
EE103	Electronics (II)
EE104	Communication Networks
ENGL101	English Grammar
ENGL102	English Writing
ENGL103	Technical Writing
MATH101	Introduction to Mathematics
MATH102	Differential Equations
MATH103	Calculus (I)
MATH104	Calculus (II)
MATH106	Algebra
MATH107	Computer Mathematics
PHYS101	Physics (I)
PHYS102	Physics (II)
STAT101	Introduction to Statistics
STAT102	Advanced Statistics

كما يُمكِن استخدامُ الأسماء المُستَعارة للجداول أيضاً في تعليمة الاختيار، وخاصةً عندما يُستخدَم الجدول نفسه أكثر من مرة في نفس التعليمة لإجراء عملية ربط (Join Operation) كما

سنرى لاحقاً عند شرح عملية الربط. والمثال التالي يوضِّح طريقة استخدام الأسماء المستعارة للجداول.

```
SELECT C.Course_Id, C.Title
FROM COURSE_T  C;
```

ويُلاحَظ في المثال السابق أنه قد تمَّ استخدامُ اسم مُستعار لجدول المواد الدراسية (COURSE_T) في عبارة مصدر الاختيار (FROM)؛ ليصبح اسمه "C". وبعد ذلك؛ تمَّ استخدام المُسمَّى المستعار للجدول ضِمن عبارة الاختيار (SELECT) في تعليمة الاختيار. وعلى الرغم من أن عبارة مصدر الاختيار (FROM) تأتي بعد عبارة الاختيار؛ فإنَّ هذا الترتيب لا يعني أن تعليمة الاختيار تُنفَّذ في نظام قاعدة البيانات حسب تسلسل العبارات في التعليمة كما سنوضِّح لاحقاً.

4-1-2-7 اختيار كافة أعمدة جدول:

لاختيار جميع أعمدة جدول ما يُمكِن استخدامُ علامة النجمة (*) مع تعليمة الاختيار. وتقدِّم هذه الطريقة اختصاراً يُعفينا من سَرْد أسماء كلِّ حقول الجدول. والشكل التالي يُمثِّل تعليمة الاختيار عند استخدام هذه الطريقة:

```
SELECT *
FROM TableName;
```

أو

```
SELECT DISTINCT
*
FROM TableName;
```

- مثـــال 5: ما تفاصيل جميع البرامج التدريسية التي توفرها الجامعة الأهلية؟

الحل:

```
SELECT *
FROM COURSE_T;
```

وتكون نتيجة التعليمة السَّابقة، كما يلي:

```
COURSE_ TITLE                                    UNITS DEPART
------- ----------------------------------- ---------- ------
CHEM101 CHEMISTRY (I)                                3 CHEM
CHEM102 CHEMISTRY (II)                               3 CHEM
CS101   JAVA PROGRAMMING                             3 CS
CS102   SOFTWARE ENGINEERING                         3 CS
CS103   C/C++ PROGRAMMING                            3 CS
CS104   COMPUTER ARCHITECTURE                        3 CS
CS105   INTRODUCTION TO DATABASE SYSTEMS             3 CS
EE101   ELECTRIC CIRCUITS                            3 EE
EE102   ELECTRONICS (I)                              3 EE
EE103   ELECTRONICS (II)                             3 EE
EE104   COMMUNICATION NETWORKS                       4 EE
ENGL101 ENGLISH GRAMMAR                              2 ENGL
ENGL102 ENGLISH WRITING                              3 ENGL
ENGL103 TECHNICAL WRITING                            3 ENGL
MATH101 INTRODUCTION To MATHEMATICS                  3 MATH
MATH102 DIFFERENTIAL EQUATIONS                       3 MATH
MATH103 CALCULUS (I)                                 3 MATH
MATH104 CALCULUS (II)                                3 MATH
MATH106 ALGEBRA                                      4 MATH
MATH107 COMPUTER MATHEMATICS                         3 MATH
PHYS101 PHYSICS (I)                                  3 PHYS
PHYS102 PHYSICS (II)                                 3 PHYS
STAT101 INTRODUCTION TO STATISTICS                   3 STAT
STAT102 ADVANCED STATISTICS                          3 STAT
```

أمَّا في حالة الرَّغبة في ترتيب حقول الجدول ترتيباً معيناً؛ فإنه لا بد من سرد أسماء حقول الجدول كافةً ضِمن تعليمة الاختيار وفقاً للترتيب الذي نرغب فيه؛ لأن استخدامَ علامة النجمة ضِمن تعليمة الاختيار تسترجع بيانات الحقول وفق الترتيب الذي عُرِّفت فيه ضِمن تعليمة إنشاء الجدول. فعلى سبيل المثال: لو أردنا استرجاع بيانات جدول المواد الدراسية وفق الترتيب (Course_ID, Units, Department_ID, Title)؛ فإنه يمكِن استخدام تعليمة الاختيار، كما يلي:

```
SELECT Course_ID, Units,
Department_ID, Title
FROM Course_T;
```

وتكون نتيجة التعليمة السابقة، كما يلي:

```
COURSE_      UNITS DEPART TITLE
------- ---------- ------ ----------------------------------
CHEM101          3 CHEM   CHEMISTRY (I)
CHEM102          3 CHEM   CHEMISTRY (II)
CS101            3 CS     JAVA PROGRAMMING
CS102            3 CS     SOFTWARE ENGINEERING
CS103            3 CS     C/C++ PROGRAMMING
CS104            3 CS     COMPUTER ARCHITECTURE
CS105            3 CS     INTRODUCTION TO DATABASE SYSTEMS
EE101            3 EE     ELECTRIC CIRCUITS
EE102            3 EE     ELECTRONICS (I)
EE103            3 EE     ELECTRONICS (II)
EE104            4 EE     COMMUNICATION NETWORKS
ENGL101          2 ENGL   ENGLISH GRAMMAR
ENGL102          3 ENGL   ENGLISH WRITING
ENGL103          3 ENGL   TECHNICAL WRITING
MATH101          3 MATH   INTRODUCTION To MATHEMATICS
MATH102          3 MATH   DIFFERENTIAL EQUATIONS
MATH103          3 MATH   CALCULUS (I)
MATH104          3 MATH   CALCULUS (II)
MATH106          4 MATH   ALGEBRA
MATH107          3 MATH   COMPUTER MATHEMATICS
PHYS101          3 PHYS   PHYSICS (I)
PHYS102          3 PHYS   PHYSICS (II)
STAT101          3 STAT   INTRODUCTION TO STATISTICS
STAT102          3 STAT   ADVANCED STATISTICS
```

7-2-1-5 الاسترجاع المشروط (**Conditional Retrieval**):

تُسترجَع البيانات عادةً باستخدام تعليمة الاختيار وفق شروط مُحدَّدة. وتوفر لغة الاستفسار البنائية العديدَ من عوامل المقارنة المنطقية التي يمكِن استخدامها ضِمن عبارة شرط الاسترجاع (WHERE). ومن عوامل المقارنة المنطقية المُستخدَمة في لغة الاستفسار البنائية، ما يلي:

عوامل المقارنة المستخدمة فى شرط الاسترجاع	
يساوى	=
لا يساوى	< >
أكبر من	>
أكبر من أو يساوى	>=
أقل من	<
أقل من أو يساوى	<=

وتأتي عبارة شرط الاسترجاع (WHERE) بعد عبارة مصدر الاسترجاع (FROM) في الترتيب ضِمن تعليمة الاختيار. والشكل التالي يوضِّح عبارة شرط الاسترجاع ضِمن تعليمة

الاختيار.

```
SELECT [*] | ColumnName1, ColumnName2, …. ColumnNameN
FROM TableName
WHERE [NOT] ColumnName  Comparison-Operator  Literal;
```

ويُقصَد بعامل المقارنة (Comparison_Operator) في تعليمة الاختيار السابقة أحد عوامل المقارنة المُدرَجة في الجدول أعلاه. أمَّا كلمة النفي (NOT)؛ فهي اختيارية، ويعني بها نفي شرط الاختيار الذي يليها، على حين يُقصَد بثابت المقارنة (Literal): إمَّا قيمة ثابتة وإمَّا مُسمَّى حقل آخر. وعادةً ما تتطلب قواعد المقارنة المنطقية توافق أنواع البيانات للحقول التي تقوم بالمقارنة بينها: مثل مقارنة رقمي برقمي أو سلسلة حرفية بسلسلة حرفية. كما يُلاحَظ أن البيانات الرقمية من نوع البيانات «SMALLINT» متوافقٌ مع نوع البيانات «NUMBER»، والسلاسل الحرفية من نوع بيانات «CHAR» متوافقٌ مع نوع البيانات «VARCHAR» أيضاً. وبذلك يعتمد نوع بيانات ثابت المقارنة على نوع بيانات العمود الذي تتمُّ معه المقارنة. وفي حال كان ثابت المقارنة ثابتاً حرفياً؛ فإنه يُوضَع بين علامتي تنصيص مفردتين () إذا كانت بيانات العمود الخاضع للمقارنة حرفياً. كما يُمكِن أيضاً استخدام اسم عمود آخر أو تعبير حسابي بدلاً من استخدام قيمة ثابتة لثابت المقارنة كما سيتضح فيما بعد. وتُمكِّن لغة الاستفسار البنائية من الربط بين أكثر من شرط استرجاع باستخدام العوامل المنطقية «و» (AND) و «أو» (OR).

- مثـــال 6: ما أسماء أعضاء هيئة التدريس الذين يعملون في قسم اللغة الإنجليزية (ENGL)؟

الحل: تُطبق تعليمة الاختيار على جدول أعضاء هيئة التدريس؛ بحيث يتمُّ اختيار حقل الاسم الأول واسم العائلة من حقول الجدول، وبحيث يُطبق شرط الاختيار على حقل رمز القسم الذي يجب أن يكون مساوياً للقيمة الثابتة (’ENGL‘).

```
SELECT FName, LName
FROM FACULTY_T
WHERE Department_ID = ‘ENGL’;
```

وتكون نتيجة التعليمة السابقة الجدول التالي الذي يحتوي على أسماء أعضاء هيئة التدريس الذين يعملون في قسم اللغة الإنجليزية، كما يلي:

FNAME	LNAME
Yahya	Khorshid
Salem	Alhamad
Salman	Albassam

- مثـــال 7: ما أسماء ورواتب أعضاء هيئة التدريس الذين يتقاضون مرتبات تساوي 000.40 ألفاً أو أكثر؟

الحل:

```
SELECT FName, LName, Salary
  FROM FACULTY_T
       WHERE Salary >= 40000;
```

وتكون نتيجة التعليمة السابقة، كما يلي:

FNAME	LNAME	SALARY
Mohammed	Alhamad	44000
Ghanim	Alghanim	44500
Saleh	Alghamdi	44600
Salem	Alhamad	40000
Fahad	Alzaid	44300
Saud	Alkhalifa	44900
Sultan	Aljasir	43300
Ali	Albader	45300
Saad	Alzhrani	44200

- مثـــال 8: ما أسماء ورواتب أعضاء هيئة التدريس الذين لا يعملون في قسم الحاسب الآلي 'CS'؟

الحل:

```
SELECT FName, LName, Salary
  FROM FACULTY_T
```

```
WHERE Department_ID < > 'CS';
```

أو

```
SELECT FName, LName, Salary
FROM FACULTY_T
WHERE NOT Department_ID = 'CS';
```

وتكون نتيجة التعليمتين السابقتين، كما يلي:

FNAME	LNAME	SALARY
Khalid	Aloufi	35000
Fahad	Alhamid	25900
Ahmad	Alotaibi	33900
Saleh	Alghamdi	44600
Yahya	Khorshid	36700
Salem	Alhamad	40000
Salman	Albassam	33800
Turki	Alturki	27800
Fahad	Alzaid	44300
Saud	Alkhalifa	44900
Mahmood	Alsalem	31900
Mishal	Almazid	29800
Sultan	Aljasir	43300
Ali	Albader	45300
Saad	Alzhrani	44200
Ahmad	Alsabti	33900

- مثـــال 9: ما أسماء ورواتب أعضاء هيئة التدريس الذين يعملون في قسم الحاسب الآلي، وتزيد رواتبهم عن 40,000 أو تقل عن 30,000؟

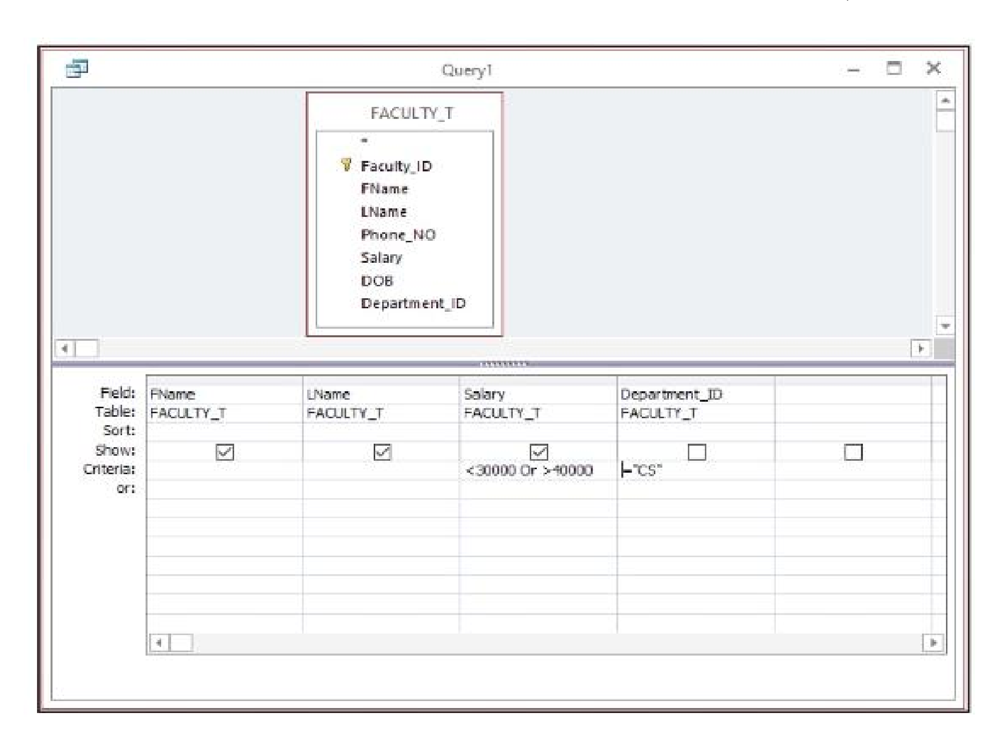
الحل:

```
SELECT FName, LName, Salary
FROM FACULTY_T
WHERE Department_ID = 'CS' AND (Salary > 40000 OR Salary < 30000);
```

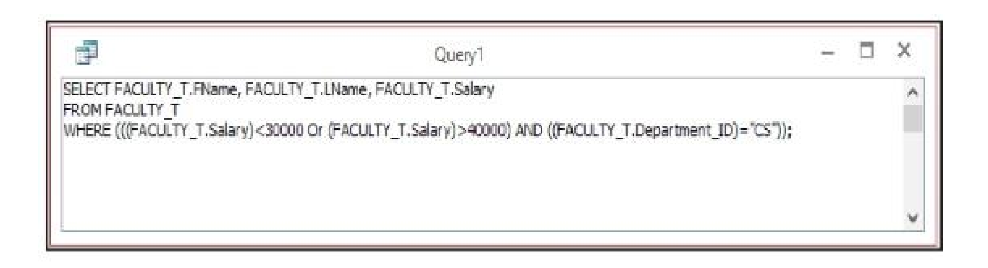
وتكون نتيجة التعليمة السابقة، كما يلي:

FNAME	LNAME	SALARY
Mohammed	Alhamad	44000
Ghanim	Alghanim	44500
Ibraheem	Alsaleh	25000

كما يُمكِن حلُّ المثال السابق في بيئة أكسس من خلال وَضْع شروط الاسترجاع في الصف المعنون بالمعيار (Criteria)، والصف الذي يليه والمعنون (OR)؛ حيث يمكِن وَضْع الشروط في هذين الصفين حسب الروابط المنطقية بين الشروط المختلفة، سواء كانت «و» أم «أو». فإذا أردنا الرَّبط بين الشروط بالرابط المنطقي «و»؛ فإننا نضع الشروط في الصف نفسه المخصَّص للمعيار (Criteria). أمَّا إذا أردنا أن نربط بين الشروط بالرابط المنطقي «أو»؛ فإننا نضعها في الصف المعنون (OR). ويمكن الربط بتوليفات مختلفة من العوامل المنطقية بين الشروط المختلفة وفق ذلك كما يوضِّح هذا المثال. كما يوضح هذا المثال أيضاً استخدام الحقل (Department_ID) من جدول أعضاء هيئة التدريس، ووَضْع شرط عليه دون إظهاره ضِمن نتيجة التعليمة؛ وذلك من خلال عدم اختياره بكلمة إظهار (Show). ويعني هذا أن هذا الحقل لن يظهر ضِمن أعمدة نتيجة العملية.



ويمكِن الانتقالُ لشاشة عرض (SQL) لرؤية تعليمة لغة الاستفسار البنائية التي تتوافق مع تصميم التعليمة السابق، والتي تكون، كما يلي:



```
SELECT FACULTY_T.FName, FACULTY_T.LName, FACULTY_T.Salary
FROM FACULTY_T
WHERE (((FACULTY_T.Salary)<30000 Or (FACULTY_T.Salary)>40000) AND ((FACULTY_T.Department_ID)="CS"));
```

7-2-1-5-1 العوامل العلاقية (**LIKE**, **BETWEEN and IN**):

يستعرضُ هذا الجزء من الفصل ثلاثة عوامل علاقية أخرى يمكِن استخدامها في شرط الاسترجاع (WHERE) هي العامل العلاقي «مثل» (LIKE) الذي يُستخدَم لتحديد مواصفات معينة للسلسلة الحرفية التي يجب أن تنطبق عليها قيمة حقل حرفي ما ضِمن عبارة شرط الاسترجاع؛ حتى يتمَّ اختيار السجل الذي ينتمي إليه الحقل ضِمن نتيجة تعليمة الاختيار، والعامل «بين» (BETWEEN) الذي يُحدِّد مدًى معيناً لقيمة حقل ما ضِمن عبارة شرط الاسترجاع؛ بحيث يجب أن تكون قيمة الحقل ضِمن المدى الذي تمَّ تحديده حتى يظهر السجل الذي ينتمي إليه الحقل ضِمن نتيجة تعليمة الاختيار، والعامل «في» (IN) الذي يُحدِّد مجموعةً من القيم التي يجب أن تكون قيمة الحقل من ضِمنها حتى يظهر السجل الذي يتبعه الحقل ضمن نتيجة تعليمة الاختيار.

7-2-1-5-1-1 العامل العلاقي «مثل» (**LIKE Comparison Operator**):

يُستخدَم العامل العلاقي «مثل» (LIKE) ضمن شرط الاسترجاع (WHERE)؛ للمقارنة بين قيمة حقلٍ ما بياناته من نوع السلاسل الحرفية بقيمة جزئية ثابتة لسلسلة حرفية ما؛ بحيث يجب أن تتضمَّن قيمة الحقل قيمة السِّلسلة الحرفية الجزئية الثابتة حتى يتحقق شرط الاختيار، ويظهر السجل الذي يتبعه الحقل ضِمن نتيجة تعليمة الاختيار. ويمكِن أن تتكون السلسلة الحرفية الجزئية من أيِّ حروف وأرقام وحروف خاصة، هي: (!#@$^&*-+=)؛ بالإضافة إلى الرموز الشاملة (WildCards) المُستخدَمة مع LIKE، وهي:

- الرمز الشامل (%): ويُستخدَم ليحل محلَّ صفر أو أكثر من الحروف في السلسلة الحرفية الجزئية. فمثلاً عند البحث عن الأسماء التي تبدأ بالحرف (G)، وتنتهي بالحرف (m) بغض النظر عن الحروف بينهما يمكِن كتابة السلسلة الحرفية الجزئية، كما يلي: (’m%G‘ LIKE). أما إذا أردنا

البحث عن الأسماء التي تبدأ بالحرف (A) بغض النظر عما يليها من حروف؛ فإنه يمكِن كتابة السلسلة الجزئية، كما يلي: (LIKE 'A%'). وإذا ما أردنا تلك التي تنتهي بالحرف (m) بغض النظر عمَّا يسبقها من حروف، يمكن كتابة السلسلة الحرفية، كما يلي: (LIKE '%m').

- الرمز الشامل (_): ويُستخدَم ليحل محلَّ حرف واحد فقط من الحروف في السلسلة الحرفية الجزئية وفي مكان مُحدَّد فيها. فعلى سبيل المثال: يمكن البحث عن الأسماء ذات طول ستة أحرف وتنتهي بالحرف (m)، كما يلي: (LIKE '_ _ _ _ _m')؛ بحيث إن تكرار الرمز الشامل في السلسلة الحرفية الجزئية السابقة؛ هو خمس مرات.

ويُمكِن استخدام الرمزين الشاملين السابقين في توليفات مختلفة ضِمن عامل المقارنة «مثل» (LIKE). فمثلاً عند البحث عن الأسماء التي تبدأ بالحرف (A) متبوعاً بحرف واحد فقط في الموقع الثاني بغض النظر عن الحرف الثاني، وبحيث يكون فيها الحرف الثالث هو (m)، وبغض النظر عمَّا يتبع الحرف الثالث من حروف؛ فإنه يمكن كتابة السلسلة الحرفية الجزئية، كما يلي: (LIKE 'A_m%'). وتجدر الملاحظة بأن حجم الحرف المُستخدَم، سواء كان صغيراً أم كبيراً، ضِمن أية سلسلة حرفية في لغة الاستفسار البنائية له أهميته. فعلى سبيل المثال: يُعَدُّ الحرف (A) والحرف (a) مختلفين نهائياً، وليس بينهما أية علاقة عند استخدامنا لهما ضِمن السلاسل الحرفية، سواء كانت ضمن عامل المقارنة «مثل» أو في أي موقع تُستخدَم فيه السلاسل الحرفية.

ويُمكِن استخدام عامل النفي (NOT) بالإضافة إلى عامل المقارنة «مثل»؛ بحيث يتمُّ اختيار جميع السجلات التي لا ينطبق عليها عامل المقارنة «مثل». وفيما يلي الشكل العام لعامل المقارنة «مثل».

```
SELECT ColumnName(s)
FROM TableName
WHERE ColumnName [NOT] LIKE
MatChing_sub_String;
```

مثال: ما أسماء أعضاء هيئة التدريس التي فيها الحرفان (ha) في الموقع الثالث والرابع من أسماء عائلاتهم، على التوالي، بغض النظر عما يسبق ذلك من حروف أو يتبعه؟

الحل:

```
SELECT FName, LName
FROM FACULTY_T
WHERE LName LIKE '__ha%';
```

وتكون نتيجة التعليمة، كما يلي:

```
FNAME        LNAME
------------ ------------
Fahad        Alhamid
Mohammed     Alhamad
Salem        Alhamad
```

7-2-1-5-1-2 العامل العلاقي «بين» (BETWEEN Comparison Operator):

يُستخدَم العامل العلاقي «بين» (BETWEEN) ضِمن شرط الاسترجاع (WHERE)؛ لتحديد مدى معين لقيمة حقلٍ ما ضِمن عبارة شرط الاسترجاع؛ بحيث يجب أن تكون قيمة الحقل ضمن المدى الذي تمَّ تحديده حتى يظهر السجل الذي ينتمي إليه الحقل ضِمن نتيجة تعليمة الاختيار. ويمكِن استخدام العامل العلاقي «بين» مع أيِّ حقلٍ، سواء كانت بياناته من النوع الرقمي، أو الحرفي، أو من نوع التاريخ والوقت. كما يمكِن استخدام عامل النفي (NOT)؛ إضافةً إلى عامل المقارنة «بين» عند الرغبة في اختيار جميع السجلات التي لا ينطبق عليها عامل المقارنة «بـين». وفيما يلي الشكل العام لعامل المقارنة «بـين».

```
SELECT ColumnName(s)
FROM TableName
WHERE ColumnName [NOT] BETWEEN Value1
AND Value2;
```

مثال: ما أسماء أعضاء هيئة التدريس الذين مرتباتهم بين 30 ألفاً و40 ألفاً؟

الحل:

```
SELECT FName, LName, Salary

FROM FACULTY_T

WHERE Salary BETWEEN 30000 AND 40000;
```

وتكون نتيجة التعليمة، كما يلي:

FNAME	LNAME	SALARY
Khalid	Aloufi	35000
Saleh	Aleesa	30000
Ahmad	Alotaibi	33900
Yahya	Khorshid	36700
Salem	Alhamad	40000
Salman	Albassam	33800
Mahmood	Alsalem	31900
Ahmad	Alsabti	33900

مثال: ما أسماء أعضاء هيئة التدريس الذين مرتباتهم أقل من 30 ألفاً أو أكثر من 40 ألفاً؟

الحل:

```
SELECT FName, LName, Salary

FROM FACULTY_T

WHERE Salary NOT BETWEEN 30000 AND 40000;
```

وتكون النتيجة، كما يلي:

FNAME	LNAME	SALARY
Fahad	Alhamid	25900
Mohammed	Alhamad	44000
Ghanim	Alghanim	44500
Ibraheem	Alsaleh	25000
Saleh	Alghamdi	44600
Turki	Alturki	27800
Fahad	Alzaid	44300
Saud	Alkhalifa	44900
Mishal	Almazid	29800
Sultan	Aljasir	43300
Ali	Albader	45300
Saad	Alzhrani	44200

مثال: ما أسماء أعضاء هيئة التدريس الذين تبدأ أسماؤهم الأولى بالحروف التي تقع بين الحرف (A) والحرف (G)؟

الحل:

```
SELECT FName, LName, Salary
FROM FACULTY_T
WHERE FName BETWEEN 'A' AND 'G';
```

وتكون النتيجة، كما يلي:

FNAME	LNAME	SALARY
Fahad	Alhamid	25900
Ahmad	Alotaibi	33900
Fahad	Alzaid	44300
Ali	Albader	45300
Ahmad	Alsabti	33900

ويُلاحَظ في النتيجة السابقة أنها لا تتضمَّن عضو هيئة التدريس «غانم» (Ghanim)؛ وذلك لكون هذا الاسم يأتي في الترتيب بعد الحرف (G) الذي تمَّ تحديده في شرط الاسترجاع. وإذا أردنا

إظهار الأسماء التي تبدأ بالحرف "G" أيضاً؛ فيمكننا تحديد الحرف التالي للحرف (G) وهو الحرف (H).

مما سبق يتضح أن العامل العلاقي «بين» يكافئ (≡) استخدام عامل المقارنة (>=) وعامل المقارنة (<=)؛ وذلك عند ربطهما بالعامل المنطقي «و» (AND) ضمن شرط الاسترجاع، كما يلي:

```
ColumnName BETWEEN Value1 AND Value2 ≡
ColumnName >= Value1 AND   ColumnName <= Value2
```

ويُمكِن حلُّ المثال السابق باستخدام عامل المقارنة «أكبر من أو يساوي» وعامل المقارنة «أصغر من أو يساوي»، كما يلي:

```
SELECT FName, LName, Salary
FROM FACULTY_T
WHERE FName >= 'A' AND FName <= 'G';
```

7-2-1-5-1-3 العامل العلاقي «في» (IN Comparison Operator):

يُستخدَم العامل العلاقي «في» (IN) ضِمن شرط الاسترجاع (WHERE)؛ لتحديد مجموعة من القِيم؛ بحيث يجب أن تكون قيمة الحقل من ضِمنها حتى يظهر السجل الذي يتبعه الحقل ضِمن نتيجة تعليمة الاختيار. وقد تكون مجموعة القيم ثابتةً أو ناتجة من خلال عمليات اختيار متداخلة أخرى كما سنوضِّح في الجزء (8-2-1). وكما هو الحال في العامل العلاقي «بين»، يمكِن أن تكون بيانات الحقل من النوع الرقمي، أو الحرفي، أو نوع التاريخ والوقت. كما يمكِن استخدام عامل النفي (NOT)؛ بالإضافة لعامل المقارنة «في» عند الرغبة في اختيار جميع السجلات التي لا ينطبق عليها عامل المقارنة «في». وفيما يلي الشكل العام لعامل المقارنة «في».

```
SELECT ColumnName(s)
```

```
FROM TableName
WHERE ColumnName [NOT] IN (Value1, Value2,
…, ValueN);
```

مثال: ما أسماء أعضاء هيئة التدريس الذين رواتبهم 40 ألفاً أو 44 ألفاً؟

الحل:

```
SELECT FName, LName, Salary
FROM FACULTY_T
        WHERE Salary IN (40000, 44000);
```

وتكون النتيجة، كما يلي:

```
FNAME        LNAME            SALARY
------------ ------------ ----------
Mohammed     Alhamad           44000
Salem        Alhamad           40000
```

- مثال: ما أسماء ورواتب أعضاء هيئة التدريس ذوي أسماء العائلات غير (Alhamid)، و(Alotaibi)، و(Alzaid)، و(Albader)، و(Alsabti).

الحل:

```
SELECT FName, LName, Salary
FROM FACULTY_T
WHERE LName NOT IN ('Alhamid', 'Alotaibi', 'Alzaid',
'Albader', 'Alsabti');
```

وتكون النتيجة، كما يلي:

```
FNAME        LNAME            SALARY
------------ ------------ ----------
Khalid       Aloufi            35000
Saleh        Aleesa            30000
Mohammed     Alhamad           44000
Ghanim       Alghanim          44500
Ibraheem     Alsaleh           25000
Saleh        Alghamdi          44600
Yahya        Khorshid          36700
Salem        Alhamad           40000
Salman       Albassam          33800
Turki        Alturki           27800
Saud         Alkhalifa         44900
Mahmood      Alsalem           31900
Mishal       Almazid           29800
Sultan       Aljasir           43300
Saad         Alzhrani          44200
```

وكما هو الحال بالنسبة للعامل العلاقي «بين»، يمكِن استخدام عوامل مقارنة أخرى؛ للحصول على ما يكافئ عامل المقارنة «في». وعلى وجه التحديد يمكِن استخدام عامل المقارنة (=) أكثر من مرة مربوطاً بالعامل المنطقي «أو» (OR) ضِمن شرط الاسترجاع، كما يلي:

```
ColumnName IN (Value1, Value2, ..., ValueN) ≡
ColumnName = Value1OR ColumnName = Value2 . . . OR ColumnName = ValueN
```

ويمكِن حلُّ المثال السابق باستخدام عامل المقارنة «يساوي» وعامل المقارنة «أو»، كما يلي:

```
SELECT FName, LName, Salary
FROM FACULTY_T
WHERE NOT (LName = 'Alhamid' OR LName = 'Alotaibi'
                OR LName = 'Alzaid' OR LName = 'Albader'
                OR LName = 'Alsabti');
```

2-5-1-2-7 القيم الغائبة (NULL Values):

تسمح لغة الاستفسار البنائية للحقول بأن تكون قيمُ بياناتها، وليس نوعية بياناتها، غائبةً (NULL). ويمكن أن تُفسَّر القيمة الغائبة (NULL) من حقلٍ ما وفق عددٍ من التفسيرات التي يأتي من ضِمنها التفسيرات الثلاثة التالية التي تُعَدُّ الأكثر شيوعاً.

- قيمة الحقل غير متوافرة حالياً (Unavailable): القيمة موجودة فعلياً على أرض الواقع؛ ولكنها غير متوفرة حالياً، ومن ثم لا يُمكِن إدخالها إلى الحقل. ومن الأمثلة القابلة لهذا التفسير قيم حقل تاريخ الميلاد؛ إذ إن لكلِّ شخص تاريخ ميلاد، وإن غياب قيمة تاريخ الميلاد لشخصٍ معين من حقل تاريخ الميلاد لا يعني أنه لا يُوجَد لهذا الشخص تاريخ ميلاد، ولكن القيمة الغائبة في هذه الحالة تعني عدم توفر تاريخ الميلاد لهذا الشخص في الوقت الراهن ضِمن بيانات قاعدة البيانات.

- قيمة الحقل مُتحفَّظ عليها (Withheld): القيمة موجودة فعلياً على أرض الواقع؛ ولكنه تم التحفظ عليها. ومن أمثلة ذلك عدم تزويد أحد الموظفين لرقم هاتفه المنزلي بشكلٍ مقصود على الرغم من وجود خط هاتف منزلي له؛ وذلك بغية التحفظ على رقم هاتفه وعدم الرغبة في إفشائه.

- قيمة الحقل غير منطبقة (InappliCable): لا تُوجَد قيمة يُمكِن إدخالها إلى الحقل، وتنطبق مع ما يتوفر على أرض الواقع. ومن أمثلة ذلك عدم توفُّر درجة علمية (بكالوريوس، ماجستير، دكتوراه) لأحد الموظفين؛ بحيث يمكن إدخالها في حقل الدرجة العلمية التابع لجدول الموظفين.

إن مبدأ القيم الغائبة مبدأ فعَّال جداً؛ إذ إنه يُعفينا من تدوين الحالات السابقة بشكلٍ واضح ضِمن بيانات قاعدة البيانات. فعلى سبيل المثال: بدون هذا المبدأ؛ فإنه يجب إدخال رمز خاص في حقل تاريخ الميلاد؛ لبيان أن تاريخ الميلاد غير متوفر حالياً مثل الرمز (12-12-1212)، أو رمز خاص لأرقام الهواتف المتحفظ عليها مثل (999-9999)، أو رمز خاص لحقل الدرجة العلمية غير المنطبق على الموظفين مثل: (no_degree). إن إدخال مثل هذه الرموز ضِمن بيانات قاعدة البيانات يُعقِّد فهم محتويات قاعدة البيانات كما أنه يؤدي إلى صعوبة التعامل معها؛ حيث يجب على جميع المستفيدين من قاعدة البيانات معرفة هذه الرموز ومعانيها. كما يعني هذا تعقيد كتابة التطبيقات المبنيَّة على قاعدة البيانات.

وعلى الرغم من فاعلية مبدأ القيم الغائبة؛ فإنه يجب توخِّي الحذر الشديد عند التعامل معها من خلال لغة الاستفسار البنائية؛ لأنها تُعَدُّ مصدراً للالتباس (GarCia-Molina et al, 2014). فعند استخدام أحد الحقول التي قد لا تحتوي على قِيم ضِمن شرط الاختيار (WHERE)؛ فإنه يجب تذكر قاعدتين رئيسيتين:

- عندما نتعاملُ مع قيمة غائبة من خلال العوامل الحسابية، مثل: الضرب والجمع؛ فإن نتيجة العملية الحسابية تكون قيمة غائبة أيضاً. فعلى سبيل المثال: لنفترض الحقل (X) التابع لأحد الجداول ونوع بياناته عددية (Number). ولنفترض أيضاً أن قيمة الحقل (X) في أحد سجلات الجدول غائبة (NULL). فإذا ما أجرينا العملية الحسابية (X+10)؛ فإن نتيجة العملية ستكون غائبة (NULL). كذلك هو الحال لو أجرينا العملية (X*0)، أو العملية (X-X)؛ فإن النتيجة ستكون غائبة أيضاً على الرغم من أنَّ ناتج عملية ضرب أيِّ عدد بـالعدد صفر هو صفر، وأن ناتج طرح أيِّ عدد من نفسه هو صفر أيضاً! ويعني هذا أن نتيجة أية عملية حسابية تتضمَّن حقلاً قيمته غائبة ستكون نتيجتها غائبـة.

- عندما نتعامل مع قيمة غائبة من خلال عوامل المقارنة مثل (=، <>، ... إلخ)؛ فإن النتيجة تكون «غير معلومة» (UNKNOWN)، (وليست «غائبة» (NULL)). فعلى سبيل المثال: عندما نقارن حقل قيمته غائبة وليكن (X) بالعدد 3 مثل (X=3) أو (X>3)؛ فإن نتيجة عملية المقارنة ستكون «غير معلومة» (UNKNOWN).

وحيث إنَّ عمليات المقارنة قد تقود إلى نتيجة «غير معلومة» (UNKNOWN)؛ فإن هذا يستدعي تعريفاً للمنطق الثلاثي القيم الذي يحتوي على القيمة «غير المعلومة» (UNKNOWN)؛ بالإضافة «للصح» (True)، و «الخطأ» (False)؛ وذلك حتى نتمكَّن من الربط بين هذه القيم المنطقية؛ من خلال العوامل المنطقية (LogiCal Operators)، وهي: «و» (AND)، و «أو» (OR)، و «النفي» (NOT).

7–2–1–5–2–1 المنطق الثلاثي القِيم (Three–Valued LogiC):

عند تعاملنا مع القِيم الغائبة؛ من خلال عوامل المقارنة قد ينتج عن ذلك قيمٌ «غير معلومة» (UNKNOWN)، كما رأينا أعلاه. عندئذٍ علينا التعاملُ مع ثلاث قيم، هي: «الصحيح»، و«الخطأ»، و«القيمة غير المعلومة». وفي ظل وجود القيمة «غير المعلومة»، علينا تفهُّم طريقة عمل العوامل المنطقية (AND, OR, NOT) في ظل وجود هذه القيمة المنطقية الجديدة.

إن القاعدة الرئيسية للتعامل مع هذه القيمة المنطقية الجديدة بسيطة جداً عندما نفكِّر بأن قيمة «الصح» هي واحد (1) وأن قيمة «الخطأ» هي صفر (0) وأن قيمة «غير المعلوم» هي النصف (½) (أيْ: إنها قيمة ما بين الصحيح والخطأ). وبناءً على ذلك يُمكِننا إعادة تعريف العوامل المنطقية، كما يلي:

1- نتيجة العامل المنطقي «و» (AND) لأيَّة قيمتين منطقيتين هي القيمة الدنيا للقيمتين المنطقيتين اللتين يربط بينهما؛ بمعنى أن (X AND Y) ستكون «خطأ» (0) إذا كان أيٌّ منهما «خطأ» (0) بغض النظر عن قيمة الآخر. أمَّا إذا كانت قيمة أيٍّ منهما «غير معلومة» (½) وليس أيٌّ منهما «خطأ» (0)؛ فإن النتيجة ستكون «غير معلومة» (½). أما إذا كان كلاهما «صح» (1)؛ فإنَّ النتيجة ستكون «صح» (1). ويعني هذا مرةً أخرى أن ناتج العامل المنطقي «و» الذي يربط بين أيِّ قيمتين منطقيتين هو قيمة الدنيا منهما.

2- نتيجة العامل المنطقي «أو» (OR) لأيِّ قيمتين منطقيتين؛ هي القيمة العليا للقيمتين المنطقيتين اللتين يربط بينهما بمعنى أن (X OR Y) ستكون «صح» (1) إذا كان أيٌّ منهما «صح» (1)، و«غير معلومة» (½) إذا لم يكن أيٌّ منهما «صح» (1) ويُوجَد على الأقل واحدة منهما «غير معلومة» (½). أمَّا إذا كان كلاهما «خطأ» (0)؛ فإن نتيجة العامل المنطقي «خطأ» (0).

3- نفي أيِّ قيمة منطقية (v)، هو (1-v). ويعني هذا أن نفي قيمة (X) عندما تكون صح (1) هو (1- قيمة X) وهي صفر، أي خطأ (0)، في هذه الحالة. أما نفي قيمة (X) عندما تكون خطأ (0)؛ فهي (1- قيمة X) وهي واحد، أي صح (1)، في هذه الحالة. أمَّا نفي قيمة (X) عندما تكون غير معلومة (½) فهي (1- قيمة X) وهي (½)، أي قيمة غير معلومة (½)، في هذه الحالة.

وتلخِّص الجداول الثلاثة التالية القيم المنطقية عند تطبيق العوامل المنطقية الثلاثة (AND, OR, NOT).

AND	T	F	U
T	T	F	U
F	F	F	F
U	U	F	U

OR	T	F	U
T	T	T	T
F	T	F	U
U	T	U	U

NOT	
T	F
F	T
U	U

عند تطبيق شرط الاسترجاع (WHERE) على حقلٍ ما ضِمن تعليمة الاختيار؛ فإن لغة الاستفسار البنائية تقوم بإظهار كلِّ سجل تكون نتيجة تطبيق شرط البحث عليه مساويةً للقيمة المنطقية الصحيحة فقط. ويعني هذا أنه إذا كانت القيمة المخزَّنة في الحقل المطبق عليه شرط الاسترجاع في أحد السجلات غائبةً، أو أن نتيجة تطبيق شرط الاسترجاع عليه هي القيمة خطأ - فإن مثل هذا السجل لن يظهر ضِمن نتيجة تعليمة الاختيار. فعلى سبيل المثال: لنفترض أننا نرغب في الاستفسار عن أعضاء هيئة التدريس الذين تزيد رواتبهم عن 40 ألفاً. عندئذٍ ستكون تعليمة لغة الاستفسار البنائية المناسبة لهذا الاستفسار، كما يلي:

```
SELECT*
FROM FACULTY_T
WHERE Salary > 40000;
```

وتكون نتيجة التعليمة السابقة عبارة عن سبعة سجلات لسبعة من أعضاء هيئة التدريس تتوافق مرتباتهم مع شرط الاسترجاع، كما يلي:

FACULTY_	FNAME	LNAME	PHONE_NO	SALARY	DOB	DEPART
320	Mohammed	Alhamad	454-5412	44000	13-MAY-65	CS
330	Ghanim	Alghanim	456-2234	44500	12-AUG-69	CS
420	Saleh	Alghamdi	454-2233	44600	13-FEB-69	CHEM
640	Fahad	Alzaid	456-3322	44300	12-MAY-71	STAT
660	Saud	Alkhalifa	454-9856	44900	13-AUG-72	STAT
770	Sultan	Aljasir	456-3212	43300	13-MAY-70	PHYS
800	Ali	Albader	456-7812	45300	22-JUN-66	EE
810	Saad	Alzhrani	454-5578	44200	17-OCT-67	EE

ولنفترض الآن أننا قمنا بتعديل راتب كلٍّ من عضو هيئة التدريس رقم (320)، وعضو هيئة التدريس رقم (810) بحيث تصبح رواتبهم غائبة (NULL). عندئذٍ ستكون نتيجة تنفيذ التعليمة، كما يلي:

FACULTY_	FNAME	LNAME	PHONE_NO	SALARY	DOB	DEPART
330	Ghanim	Alghanim	456-2234	44500	12-AUG-69	CS
420	Saleh	Alghamdi	454-2233	44600	13-FEB-69	CHEM
640	Fahad	Alzaid	456-3322	44300	12-MAY-71	STAT
660	Saud	Alkhalifa	454-9856	44900	13-AUG-72	STAT
770	Sultan	Aljasir	456-3212	43300	13-MAY-70	PHYS
800	Ali	Albader	456-7812	45300	22-JUN-66	EE

ويعني هذا أن أعضاء هيئة التدريس الذين لا ينطبق عليهم شرط الاسترجاع (رواتبهم ليست أكثر من 40 ألفاً) وأولئك الذين قِيم حقول رواتبهم غائبة لن تظهر بياناتهم ضِمن عملية الاختيار. ويعني هذا مرةً أخرى أن ما يظهر ضِمن نتائج عملية الاختيار هي تلك السجلات التي تكون فيها نتيجة شرط الاسترجاع صحيحة فقط. أمَّا تلك السجلات التي تكون فيها نتيجة شرط الاسترجاع خطأ أو غير معلومة؛ فلن تظهر من ضمن نتائج التعليمة.

أمَّا إذا أردنا استرجاع السجلات ذات القيم الغائبة، أو استرجاع السجلات ذات القِيم غير الغائبة بغضِّ النظر عما تحتويه من قيم؛ فإن لغة الاستفسار البنائية توفر معامل مقارنة خاص لهذا الغرض وهو (‘IS NULL’, ‘IS NOT NULL’)، كما يلي:

WHERE ColumnName IS NULL

أو

WHERE ColumnName IS NOT NULL

فلو أردنا استرجاع سجلات أعضاء هيئة التدريس ذوي الرواتب الغائبة، يمكِن استخدام تعليمة الاختيار التالية:

```
SELECT *
FROM FACULTY_T
WHERE Salary IS NULL;
```

وتكون نتيجة التعليمة السابقة، كما يلي:

FACULTY_	FNAME	LNAME	PHONE_NO	SALARY	DOB	DEPART
320	Mohammed	Alhamad	454-5412		13-MAY-65	CS
810	Saad	Alzhrani	454-5578		17-OCT-67	EE

أما إذا أردنا استرجاع سجلات أعضاء هيئة التدريس الذين لديهم رواتب بغض النظر عن قِيمها، يمكن استخدام التعليمة التالية:

```
SELECT *
FROM FACULTY_T
WHERE Salary IS NOT NULL;
```

وتكون نتيجة تنفيذ التعليمة السابقة، والتي لا يظهر من ضِمنها سجلات أعضاء هيئة التدريس ذوي الرواتب الغائبة، كما يلي:

FACULTY_	FNAME	LNAME	PHONE_NO	SALARY	DOB	DEPART
200	Khalid	Aloufi	454-2341	35000	22-MAY-63	MATH
220	Fahad	Alhamid	456-7733	25900	07-OCT-70	MATH
310	Saleh	Aleesa	454-8932	30000	13-SEP-66	CS
330	Ghanim	Alghanim	456-2234	44500	12-AUG-69	CS
340	Ibraheem	Alsaleh	454-1234	25000	20-JAN-70	CS
400	Ahmad	Alotaibi	454-4563	33900	17-MAY-71	CHEM
420	Saleh	Alghamdi	454-2233	44600	13-FEB-69	CHEM
500	Yahya	Khorshid	456-2221	36700	12-MAR-65	ENGL
540	Salem	Alhamad	456-3304	40000	11-SEP-72	ENGL
560	Salman	Albassam	454-7865	33800	13-SEP-68	ENGL
600	Turki	Alturki	456-7891	27800	23-JUL-75	STAT
640	Fahad	Alzaid	456-3322	44300	12-MAY-71	STAT
660	Saud	Alkhalifa	454-9856	44900	13-AUG-72	STAT
710	Mahmood	Alsalem	456-3323	31900	19-FEB-73	PHYS
730	Mishal	Almazid	454-2343	29800	17-SEP-75	PHYS
770	Sultan	Aljasir	456-3212	43300	13-MAY-70	PHYS
800	Ali	Albader	456-7812	45300	22-JUN-66	EE
850	Ahmad	Alsabti	456-0120	33900	15-APR-73	EE

ويمكِن استخدام عامل المقارنة (IS NULL) و (IS NOT NULL) مع أية توليفات أخرى من عوامل المقارنة المنطقية التي سبق التطرُّق إليها ضِمن شرط الاسترجاع.

6-1-2-7 ترتيــب نتيجة عملية الاختيار باستخدام عبارة (ORDER BY):

على الرَّغم من وجود ترتيب افتراضي للحقول ضِمن الجداول، وهو نفس ترتيبها في تعليمة الإنشاء الذي تمَّ استخدامها لإنشاء الجداول؛ فإنه لا يُوجَد ترتيب افتراضي للسجلات في أيِّ جدول. لذلك توفر لغة الاستفسار البنائية عبارة الترتيب (Order By)؛ وذلك عند الرغبة في ترتيب نتيجة عملية الاختيار وَفق قِيم حقل أو أكثر من حقول الجدول. ويمكِن ترتيب قِيَم أيِّ حقل ضِمن نتيجة العملية إمَّا بشكلٍ تصاعدي وإما بشكلٍ تنازلي. أمَّا إذا تضمَّن الحقل الذي ستُجرَى عليه عملية الترتيب قيماً غائبة؛ فإنها تظهر في نهاية النتيجة إذا كان الترتيب تصاعدياً وفي بداية النتيجة إذا كان الترتيب تنازلياً. ويعني هذا أن القيمة الغائبة من حقلٍ ما تُعَدُّ الأكبر من أيَّة قيمة مُخزَّنة في نفس الحقل؛ ولكن لسجلات أخرى في الجدول ذاته. كما أن الحالة الافتراضية للترتيب في حالة عدم تحديد طريقة الترتيب على قِيم الحقل؛ فهي الترتيب التصاعدي. لذلك يمكننا الاستغناءُ عن ذكر طريقة الترتيب في حالة رغبتنا في إجراء الترتيب التصاعدي على قِيم حقل ما. وفيما يلي الشكل العام لعبارة الترتيب في تعليمة الاختيار.

```
SELECT Column_Name(s)
```

```
FROM Tabel_Name

WHERE Condition

ORDER BY ColumnName [ASC] [, ColumnName [ DESC ]...];
```

ومثالٌ على عبارة الترتيب، لنفترض أننا نرغب في استرجاع سجلات أعضاء هيئة التدريس الذين تزيد رواتبهم عن 40 ألفاً، وترتيب النتيجة تنازلياً وفقاً للرواتب التي يتقاضونها. في هذه الحالة يمكِن استخدام تعليمة الاختيار التالية:

```
SELECT *

FROM FACULTY_T

WHERE Salary > 40000

ORDER BY Salary DESC;
```

وتكون نتيجة العملية الجدول التالي الذي يُلاحَظ فيه ترتيب السجلات تنازلياً وفقاً للرواتب التي يتقاضاها أعضاء هيئة التدريس:

FACULTY_	FNAME	LNAME	PHONE_NO	SALARY	DOB	DEPART
800	Ali	Albader	456-7812	45300	22-JUN-66	EE
660	Saud	Alkhalifa	454-9856	44900	13-AUG-72	STAT
420	Saleh	Alghamdi	454-2233	44600	13-FEB-69	CHEM
330	Ghanim	Alghanim	456-2234	44500	12-AUG-69	CS
640	Fahad	Alzaid	456-3322	44300	12-MAY-71	STAT
770	Sultan	Aljasir	456-3212	43300	13-MAY-70	PHYS

أمَّا إذا أردنا استرجاع سجلات أعضاء هيئة التدريس الذين تزيد رواتبهم عن 40 ألفاً، وأولئك ذوي الرواتب الغائبة، وترتيب النتيجة تنازلياً وفقاً للرواتب التي يتقاضونها؛ فإنه يمكِن استخدام تعليمة الاختيار التالية:

```
SELECT *
FROM FACULTY_T
WHERE Salary > 40000 OR Salary IS NULL
ORDER BY Salary DESC;
```

ويُلاحَظ في الجدول التالي الذي يُمثِّل نتيجة التعليمة السابقة أنه قد تمَّ استرجاع السجلات المطلوبة حسب شرط الاسترجاع وترتيبها تنازلياً حسب الراتب. كما يُلاحَظ ظهور سجلات أعضاء هيئة التدريس ذوي الرواتب الغائبة في بداية الترتيب.

FACULTY_	FNAME	LNAME	PHONE_NO	SALARY	DOB	DEPART
320	Mohammed	Alhamad	454-5412		13-MAY-65	CS
810	Saad	Alzhrani	454-5578		17-OCT-67	EE
800	Ali	Albader	456-7812	45300	22-JUN-66	EE
660	Saud	Alkhalifa	454-9856	44900	13-AUG-72	STAT
420	Saleh	Alghamdi	454-2233	44600	13-FEB-69	CHEM
330	Ghanim	Alghanim	456-2234	44500	12-AUG-69	CS
640	Fahad	Alzaid	456-3322	44300	12-MAY-71	STAT
770	Sultan	Aljasir	456-3212	43300	13-MAY-70	PHYS

أمَّا إذا أردنا استرجاع سِجلات أعضاء هيئة التدريس الذين تزيد رواتبهم عن 40 ألفاً، وأولئك ذوو الرواتب الغائبة، وترتيب النتيجة تصاعدياً وفقاً للرواتب التي يتقاضونها؛ فإنه يمكن استخدام تعليمة الاختيار التالية التي يُلاحَظ فيها عدم استخدام الكلمة المحجوزة (ASC)؛ لأن الترتيب يكون تصاعدياً بشكل افتراضي عند عدم تحديد ترتيب معين للنتيجة.

```
SELECT *
FROM FACULTY_T
WHERE Salary > 40000 OR Salary IS NULL
ORDER BY Salary;
```

ويُلاحَظ في الجدول التالي الذي يمثل نتيجة التعليمة السابقة أنه قد تمَّ استرجاع السجلات المطلوبة حسب شرط الاسترجاع وترتيبها تصاعدياً حسب الراتب. كما يُلاحَظ ظهور سجلات أعضاء هيئة التدريس ذوي الرواتب الغائبة في نهاية الترتيب.

FACULTY_	FNAME	LNAME	PHONE_NO	SALARY	DOB	DEPART
770	Sultan	Aljasir	456-3212	43300	13-MAY-70	PHYS
640	Fahad	Alzaid	456-3322	44300	12-MAY-71	STAT
330	Ghanim	Alghanim	456-2234	44500	12-AUG-69	CS
420	Saleh	Alghamdi	454-2233	44600	13-FEB-69	CHEM
660	Saud	Alkhalifa	454-9856	44900	13-AUG-72	STAT
800	Ali	Albader	456-7812	45300	22-JUN-66	EE
320	Mohammed	Alhamad	454-5412		13-MAY-65	CS
810	Saad	Alzhrani	454-5578		17-OCT-67	EE

ومرةً أخرى؛ نستخلص أن القيمة الغائبة تُعَدُّ العليا من حيث الترتيب بمعنى أنها تظهر في بداية النتيجة عندما يكون الترتيب تنازلياً، وفي نهاية الترتيب عندما يكون الترتيب تصاعدياً.

كما يُمكِن ترتيب نتيجة عملية الاختيار وفق قِيم أكثر من حقل كما أسلفنا سابقاً. فعلى سبيل المثال: لنفترض أننا نرغب في استرجاع سجلات أعضاء هيئة التدريس الذين تزيد رواتبهم على 40 ألفاً، وترتيب النتيجة تصاعدياً وفقاً لرموز الأقسام الدراسية التي يتبعونها، وتنازلياً وفقاً لأسماء عائلاتهم؛ بحيث يظهر اسم القسم أولاً متبوعاً باسم عضو هيئة التدريس. وللحصول على النتيجة المطلوبة يمكِن استخدام تعليمة الاختيار التالية:

```
SELECT Department_ID, LName
FROM FACULTY_T
WHERE Salary > 40000
ORDER BY Department_ID ASC, LName DESC;
```

ويُلاحَظ في الجدول التالي الذي يمثل نتيجة التعليمة السَّابقة أنه قد تمَّ استرجاع السجلات المطلوبة حسب شرط الاسترجاع (بحيث يكون الراتب أكثر من 40 ألفاً) وترتيبها تصاعدياً حسب

اسم القسم الدراسي (أولاً)، ومن ثم ترتيب أسماء أعضاء هيئة التدريس تنازلياً حسب أسماء عائلاتهم، أن قسم الإحصاء (STAT) قد جاء في نهاية الترتيب التصاعدي وفقاً لترتيب الأقسام الدراسية، وأن عضو هيئة التدريس ذا اسم العائلة «الزيد» (Alzaid)
قد جاء قبل عضو هيئة التدريس ذي اسم العائلة «الخليفة» (Alkhalifa) بشكلٍ تنازلي داخل رمز قسم الإحصاء.

1-6-1-2-7 ترتيب النتائج وفقاً للأرقام النِّسبية للأعمدة:

تُمكِّن لغة الاستفسار البنائية من ترتيب نتائج عملية الاختيار، في عبارة ترتيب نتائج عملية الاختيار (ORDER BY)، وفقاً للأرقام النسبية للحقول التي يتمُّ اختيارها ضِمن نتيجة عملية الاختيار عوضاً عن استخدام أسمائها الفعلية. ويكون شكل التعليمة في هذه الحالة، كما يلي:

```
SELECT Column_Name(s)
FROM Tabel_Name
WHERE Condition
ORDER BY ColumnNumber1 [ASC] [,
ColumnNumber2 [DESC]...];
```

فعلى سبيل المثال: يمكِن استخدامُ الأرقام النسبية للحقول عند الرغبة في إظهار عناوين المواد الدراسية التي تُنفَّذ من قِبل قسم الكيمياء وقسم الحاسب الآلي مرتبة حسب رمز القسم (أولاً) بشكلٍ تصاعدي وحسب عناوين (أو مُسمَّيات) المواد الدراسية التي ينفذها كل قسم (ثانياً) بشكلٍ تنازلي، كما يلي:

```
SELECT Department_ID, Title
FROM COURSE_T
WHERE Department_ID IN ('CS',
'CHEM')
ORDER BY 1 ASC, 2 DESC;
```

وتكون نتيجة التعليمة، كما يلي:

```
DEPART TITLE
------ -----------------------------------
CHEM   CHEMISTRY (II)
CHEM   CHEMISTRY (I)
CS     SOFTWARE ENGINEERING
CS     JAVA PROGRAMMING
CS     INTRODUCTION TO DATABASE SYSTEMS
CS     COMPUTER ARCHITECTURE
CS     C/C++ PROGRAMMING
```

كما يُمكِن استخدام الأرقام النسبية للحقول حتى لو تمَّ اختيار جميع حقول جدولٍ ما إذا ما عرف ترتيب الحقول وفق تعليمة إنشاء الجدول. فعلى سبيل المثال: لو أردنا إظهار أسماء أعضاء هيئة التدريس الذين يعملون في قسم الكيمياء وقسم الحاسب الآلي مرتبة تصاعدياً حسب أسمائهم الأولى وتنازلياً حسب رواتبهم، يمكِن استخدام تعليمة الاختيار وفق الصيغة التالية:

```
SELECT *
FROM FaCulty_T
WHERE Department_ID IN ('CS',
'CHEM')
ORDER BY 2 ASC, 5 DESC;
```

وتكون نتيجة التعليمة، كما يلي:

```
FACULTY_ FNAME        LNAME        PHONE_NO     SALARY DOB       DEPART
-------- ------------ ------------ -------- ---------- --------- ------
400      Ahmad        Alotaibi     454-4563      33900 17-MAY-71 CHEM
330      Ghanim       Alghanim     456-2234      44500 12-AUG-69 CS
340      Ibraheem     Alsaleh      454-1234      25000 20-JAN-70 CS
320      Mohammed     Alhamad      454-5412      44000 13-MAY-65 CS
420      Saleh        Alghamdi     454-2233      44600 13-FEB-69 CHEM
310      Saleh        Aleesa       454-8932      30000 13-SEP-66 CS
```

ويُلاحَظ في نتيجة التعليمة السابقة أنه قد تمَّ ترتيب النتيجة وفق الأسماء الأولى بشكل تصاعدي، ومن ثم يكون ترتيب النتيجة وفقاً للراتب بشكلٍ تنازلي. وعند تطابق الأسماء الأولى، كما في حالة الاسم «صالح» (Saleh)؛ يتمُّ الترتيب وفق الراتب بشكلٍ تنازلي، كما توضِّحه نتيجة التعليمة السابقة.

وتكمُنُ أهمية ترتيب الحقول وفقاً لأرقامها النسبية بشكلٍ خاص عند استخدام القيم المحسوبة التي لا تكون عادةً من ضِمن حقول الجدول، وإنما يتمُّ حسابها باستخدام تعليمة الاختيار وفق حقل أو أكثر من حقول الجدول، كما يوضِّح الجزء التالي من هذا الفصل.

7-1-2-7 القِيم المحسوبة (Computed Values):

يُمكِن استخدامُ التعبيرات الحسابية ضِمن تعليمة الاختيار، وتطبيقها على حقول الجدول أو استخدام الدوال الحسابية التي توفرها لغة الاستفسار البنائية؛ حيث يمكِن أن تحتوي نتيجة تعليمة الاختيار؛ إضافةً لما يتمُّ اختياره من حقول الجدول، على تعبيرات حسابية تحتوي على أسماء حقول وقيم عددية ثابتة تربطها العوامل الحسابية، وهي: الضرب (*)، والقسمة (/)، والجمع (+)، والطرح (-). وكما هو الحال في لغات برمجة الحاسب الآلي؛ يتمُّ تقييم عملية الضرب وعملية القسمة في التعبير الحسابي أولاً، ومن الجهة اليسرى للجهة اليمنى، ثم تقييم عملية الجمع وعملية الطرح ثانياً، ومن الجهة اليسرى للجهة اليمنى أيضاً. ولتغيير أولويات التقييم هذه أو إزالة الالتباس منها، يُمكِن استخدام الأقواس؛ بحيث يتمُّ تقييمها من الداخل للخارج. وعند الرغبة في ترتيب نتائج عملية الاختيار وفق قيم محسوبة، يستخدم الرقم النسبي لحقل القيمة المحسوبة حسب ترتيبه ضِمن عبارة الاختيار (SELECT).

وثمَّة مثال على القِيم المحسوبة وترتيب النتائج وفقاً لها، لنفترض أننا نرغب في إظهار أسماء أعضاء هيئة التدريس الذين يعملون في قسم الكيمياء، وقسم الحاسب الآلي، ورواتبهم بعد زيادتها بنسبة خمسة عشر في المائة (15%) مرتبة تنازلياً حسب الراتب بعد زيادته. في هذه الحالة يمكِن استخدام تعليمة الاختيار التالية:

```
SELECT FName, LName, Salary*1.15
```

```
FROM FACULTY_T
WHERE Department_ID IN ('CS', 'CHEM')
ORDER BY 3 DESC;
```

عند تنفيذ التعليمة السابقة تكون نتيجتها، كما يلي:

FNAME	LNAME	SALARY*1.15
Saleh	Alghamdi	51290
Ghanim	Alghanim	51175
Mohammed	Alhamad	50600
Ahmad	Alotaibi	38985
Saleh	Aleesa	34500
Ibraheem	Alsaleh	28750

كما يُمكِن استخدام القيم المحسوبة ضِمن عبارة شرط الاختيار (WHERE). فلو افترضنا أننا نرغب في إظهار أسماء أعضاء هيئة التدريس الذين يعملون في قسم الكيمياء وقسم الحاسب الآلي، وستزيد رواتبهم بعد نسبة الزيادة (15%) على خمسين ألفاً، يمكن استخدام تعليمة الاختيار التالية:

```
SELECT FName, LName
FROM FACULTY_T
WHERE Department_ID IN ('CS', 'CHEM')
        AND Salary*1.15 > 50000;
```

عند تنفيذ التعليمة السابقة تكون نتيجتها، كما يلي:

FNAME	LNAME
Mohammed	Alhamad
Ghanim	Alghanim
Saleh	Alghamdi

وبمقارنة نتيجة هذه التعليمة بنتيجة التعليمة السابقة يُلاحَظ أن الأسماء الأولى الثلاثة من نتيجة التعليمة السابقة هي التي ظهرت ضِمن نتيجة التعليمة الحالية؛ لكون أسماء أعضاء هيئة التدريس هذه هي من أسماء العاملين في قسم الكيمياء وقسم الحاسب الآلي، وستزيد رواتبهم على 50 ألفاً بعد زيادة الرواتب بنسبة خمسة عشر في المائة.

كما تجدرُ الإشارة هنا إلى أن القِيم المحسوبة عند استخدامها ضِمن عملية الاختيار لا تؤثر بأيِّ شكلٍ كان فيما هو مُخزَّن في حقول الجداول، خاصةً إذا تذكَّرنا أن تعليمة الاختيار ما هي إلا عملية استفسار (أو عملية استرجاع) لا تؤثر في محتويات قاعدة البيانات.

7–2–1–8 دوال التجميع (أو الأعمدة) (Aggregate (or Column) FunCtions):

توفِّر لغة الاستفسار البنائية عدداً من الدوال الإحصائية المُسمَّاة «دوال التجميع» (Aggregate FunCtions). وتُسمَّى هذه الدوال أحياناً «دوال الأعمدة» (Column FunCtions)؛ وذلك لكونها تطبق على حقول الجداول كأعمدة كاملة فيها. وهذه الدوال، هي: دالة «العدد» (COUNT)، ودالة «القيمة الصغرى» (MIN)، ودالة «القيمة الكبرى» (MAX)، ودالة «الجمع» (SUM)، ودالة «المتوسط» (AVG). وتكون نتيجة أية دالة تجميع عبارة عن صف واحد يمثل نتيجة قيمة الدالة عوضاً عن مجموعة من الصفوف تمثل محتويات الجدول الذي تمَّ تطبيق الدالة عليه.

تُستخدَم دالة العدد (COUNT) لإيجاد عدد صفوف جدولٍ ما، ينطبق عليها شرط الاختيار (WHERE) في حالة وجوده، في تعليمة الاختيار. فلمعرفة عدد أعضاء هيئة التدريس في الجامعة الأهلية؛ يمكن استخدام تعليمة الاختيار التالية:

```
SELECT COUNT(*)
FROM FACULTY_T;
```

وتكون نتيجة التعليمة السابقة؛ هي العدد عشرون (20)؛ وذلك لكون عدد أعضاء هيئة التدريس في الجامعة عشرين عضواً، كما يلي:

```
COUNT(*)
----------
        20
```

أمَّا إذا أردنا معرفة عدد أعضاء هيئة التدريس في كلٍّ من قسم الكيمياء وقسم الحاسب الآلي؛ يمكن استخدام التعليمة التالية التي تتضمَّن شرط الاختيار المناسب (وهو أن يكون القسم الذي يتبعه عضو هيئة التدريس: إما قسم الكيمياء وإما قسم الحاسب الآلي)، كما يلي:

```
SELECT COUNT(*)
FROM FACULTY_T
WHERE Department_ID IN ('CS',
'CHEM');
```

وتكون نتيجة التعليمة السابقة، كما يلي:

```
COUNT(*)
----------
         6
```

كما يمكِن أن تُستخدَم دالة العدد لإيجاد عدد صفوف نتيجة عملية الاختيار مع استبعاد الصفوف المتكررة في قيمة الحقل الذي تطبق عليه دالة العدد، وكذلك القيم الغائبة. فعلى سبيل المثال: يمكن معرفة عدد رواتب أعضاء هيئة التدريس المختلفة (أيْ: دون تكرار) باستخدام التعليمة التالية:

```
SELECT COUNT(DISTINCT Salary)
FROM FACULTY_T;
```

وتكون نتيجة التعليمة كما يلي:

```
COUNT(DISTINCTSALARY)
---------------------
                   19
```

وعلى الرغم من وجود عشرين عضواً من أعضاء هيئة التدريس في الجامعة الأهلية؛ فإن نتيجة دالة العدد لم تظهر سوى تسعة عشر (19)؛ وذلك لكون اثنين من أعضاء هيئة التدريس يتقاضون الراتب نفسه (وهو 900.33). لذلك؛ فإن دالة العدد التي طبقت على حقل الراتب قامت بعَدِّ واحدٍ من هذين الراتبين فقط مع حذف المتكرِّر منها.

أمَّا إذا استخدمنا دالة العدد وفق التعليمة التالية؛ فإنها لن تقوم بعَدِّ أعضاء هيئة التدريس الذين لم تحَدَّد رواتبهم بعد (أيْ: في حالة كون قِيم رواتبهم غائبة (NULL)) ضمن نتيجة الدالة.

```
SELECT COUNT(Salary)
FROM FACULTY_T;
```

أمَّا دالة القيمة الصغرى (MIN)؛ فتقوم بتحديد القيمة الصغرى للحقل المطبقة عليه من ضِمن الصفوف التي ينطبق عليها شرط الاختيار، ودالة القيمة الكبرى (MAX) تقوم بتحديد القيمة الكبرى للحقل المطبقة عليه من ضمن الصفوف التي ينطبق عليها شرط الاسترجاع. فعلى سبيل المثال: يمكننا معرفة أصغر راتب وأكبر راتب يتقاضاه أعضاء هيئة التدريس العاملون في قسم الكيمياء وقسم الحاسب الآلي باستخدام التعليمة التالية:

```
SELECT MIN(Salary), MAX(Salary)
FROM FACULTY_T
WHERE Department_ID IN (‘CS’, ‘CHEM’);
```

وتكون نتيجة التعليمة السابقة عبارة عن 000.25 للقيمة الصغرى (وهو أقل راتب يتقاضاه أعضاء هيئة التدريس في كلا القسمين)، و600.44 للقيمة الكبرى (وهو أعلى راتب يتقاضاه أعضاء هيئة التدريس في كلا القسمين)، كما يلي:

```
MIN(SALARY) MAX(SALARY)
----------- -----------
      25000       44600
```

تجدر الإشارة إلى أنه في حالة وجود قِيم غائبة من الحقل الذي تطبق عليه القيمة الصغرى أو القيمة الكبرى؛ فإن هذه القِيم يتمُّ تجاهلها في كلتا العمليتين ولا تدخل ضمن نتيجتيهما. كما تجدر الإشارة إلى أنه بالإمكان استخدام كلتا العمليتين مع حقول ليست من نوع الأعداد؛ إذ يمكِن استخدامهما مع السلاسل الحرفية، والتاريخ والوقت، كذلك. فمثلاً؛ يمكِننا التعرُّف على أصغر اسم عائلة (من حيث الترتيب الأبجدي) وأكبر اسم عائلة لأعضاء هيئة التدريس العاملين في قسم الكيمياء، وقسم الحاسب الآلي وفق التعليمة التالية:

```
SELECT MIN(LName), MAX(LName)
FROM FACULTY_T
WHERE Department_ID IN ('CS', 'CHEM');
```

وتكون نتيجة التعليمة أن اسم العائلة (Aleesa) هو الأصغر أبجدياً واسم العائلة (Alsaleh) هو الأكبر، كما يلي:

```
MIN(LNAME)   MAX(LNAME)
------------ ------------
Aleesa       Alsaleh
```

أما دالة المتوسط (AVG) فتُستخدَم لحساب القيمة المتوسطة للحقل المطبقة عليه في الصفوف التي ينطبق عليها شرط الاختيار مع تجاهل القِيم الغائبة من قِبل العملية الحسابية. فعلى سبيل المثال: يمكِن حساب متوسط رواتب أعضاء هيئة التدريس العاملين في الجامعة الأهلية باستخدام التعليمة التالية:

```
SELECT AVG(Salary)
FROM FACULTY_T;
```

وتكون نتيجة التعليمة، كما يلي:

```
AVG(SALARY)
-----------
      36940
```

أمَّا إذا أردنا حساب متوسط رواتب أعضاء هيئة التدريس دون اعتبار المتكرِّر منها؛ فإنه يتمُّ استخدام عبارة (DISTINCT). وفي هذه الحالة يتمُّ إدخال أيِّ قيمة متكررة مرة واحدة فقط عند حساب المتوسط، كما يلي:

```
SELECT AVG(DISTINCT Salary)
FROM FACULTY_T;
```

ونظراً لتكرار الراتب 900.33 مرتين، تكون نتيجة العملية السابقة، كما يلي:

```
AVG(DISTINCTSALARY)
-------------------
              37100
```

كما يمكِن استخدام دالة المتوسط على مجموعة جزئية من صفوف الجدول ينطبق عليها شرط الاختيار. فمثلاً: يمكِن حساب متوسط رواتب أعضاء هيئة التدريس العاملين في قسم الكيمياء وقسم الحاسب الآلي وفق التعليمة التالية:

```
SELECT AVG(Salary)
FROM FACULTY_T
WHERE Department_ID IN ('CS', 'CHEM');
```

وتكون نتيجة التعليمة السابقة، كما يلي:

```
AVG(SALARY)
-----------
      37000
```

أمَّا دالة الجمع (SUM)؛ فتُستخدَم لجَمْع قِيم الحقل المطبقة عليه في الصفوف التي ينطبق عليها شرط الاختيار مع تجاهل القِيم الغائبة من قِبل العملية الحسابية. فعلى سبيل المثال: يمكِن حساب مجموع رواتب أعضاء هيئة التدريس العاملين في الجامعة الأهلية باستخدام التعليمة التالية:

```
SELECT SUM(Salary)
FROM FACULTY_T;
```

وتكون نتيجة التعليمة، كما يلي:

```
SUM(SALARY)
-----------
     738800
```

أمَّا إذا أردنا معرفة مجموع رواتب أعضاء هيئة التدريس دون اعتبار المتكرِّر منها؛ فإنه يتمُّ استخدام عبارة (DISTINCT). وفي هذه الحالة يتمُّ إدخال أية قيمة متكررة مرةً واحدة فقط عند حساب المجموع، كما يلي:

```
SELECT SUM(DISTINCT Salary)
FROM FACULTY_T;
```

ونظراً لتكرار الراتب 900.33 مرتين، تكون نتيجة العملية السابقة، كما يلي:

```
SUM(DISTINCTSALARY)
-------------------
             704900
```

كما يُمكِن استخدام دالة الجمع على مجموعة جزئية من صفوف الجدول التي ينطبق عليها شرط الاختيار. فمثلاً؛ يمكن حساب مجموع رواتب أعضاء هيئة التدريس العاملين في قسم الكيمياء وقسم الحاسب الآلي وفق التعليمة التالية:

```
SELECT SUM(Salary)
FROM FACULTY_T
```

```
WHERE Department_ID IN ('CS', 'CHEM');
```

وتكون نتيجة التعليمة السابقة، كما يلي:

```
SUM(SALARY)
-----------
     222000
```

9-1-2-7 عبارة التجميع (GROUP BY):

في الكثيرِ من الأحيان تظهر الحاجة إلى تطبيق دوال التجميع على مجموعات جزئية من سجلات جدولٍ ما. فعلى سبيل المثال: قد نحتاج إلى معرفة متوسط، أو مجموع رواتب أعضاء هيئة التدريس في كلِّ قسم دراسي على حِدَة. في هذه الحالة؛ نحتاج إلى تقسيم سجلات الجدول إلى مجموعات غير متداخلة من السجلات؛ بحيث تحتوي كل مجموعة على السجلات التي تتوافق مع بعضها في الحقول المراد التقسيم عليها. وتُسمَّى هذه الحقول «حقول التقسيم» أو «حقول التجميع» (Grouping Attributes). بعد ذلك نقومُ باستخدام دالة التجميع المراد تطبيقها على كلِّ مجموعة بشكلٍ مستقل. وتوفر لغة الاستفسار البنائية عبارة «التجميع حسب» (GROUP BY) لهذا الغرض. وتُحدِّد عبارة «التجميع حسب» الحقول التي سيتم تقسيم الجدول وفقاً لها، والتي يجب أن تكون من ضمن الحقول التي ستظهر ضِمن نتيجة عبارة الاختيار. وبهذا الشكل سوف تظهر قيمة الحقول التي تمَّ تقسيم الجدول وفقاً لها مع قيمة دالة التجميع التي تم تطبيقها على الجدول.

فعلى سبيل المثال: يمكن معرفة عدد أعضاء هيئة التدريس، ومتوسط رواتبهم، ومجموع رواتبهم حسب الأقسام الدراسية التي يعملون فيها باستخدام التعليمة التالية:

```
SELECT Department_ID, COUNT(*),
AVG(Salary), SUM(Salary)
FROM FACULTY_T
GROUP BY Department_ID;
```

يُلاحَظ في التعليمة السابقة وجود رمز القسم الدراسي ضِمن الحقول التي تمثل نتيجة عبارة الاختيار التي يتمُّ تقسيم الجدول وفقاً لقيمته في عبارة «تجميع حسب»، الأمر الذي يُعَدُّ ضرورياً لعمل عبارة «تجميع حسب». بالإضافة إلى ذلك يظهر ضِمن عبارة الاختيار دوال التجميع الواجب تطبيقها على كلّ مجموعة من المجموعات الناتجة من عملية تقسيم الجدول. وتكون نتيجة التعليمة السابقة، التي تظهر عدد أعضاء هيئة التدريس في كلِّ قسم دراسي، ومتوسط رواتبهم حسب القسم الذي يعملون فيه، ومجموع رواتب أعضاء هيئة التدريس في كل قسم دراسي، كما يلي:

DEPART	COUNT(*)	AVG(SALARY)	SUM(SALARY)
CHEM	2	39250	78500
CS	4	35875	143500
EE	3	41133.3333	123400
ENGL	3	36833.3333	110500
MATH	2	30450	60900
PHYS	3	35000	105000
STAT	3	39000	117000

وإذا أردنا ترتيب نتائج العملية السابقة أبجدياً وبشكلٍ تنازلي حسب رموز الأقسام الدراسية؛ فإنه يُمكِن استخدام عبارة «ترتيب حسب» (ORDER BY)، كما يلي:

```
SELECT Department_ID, COUNT(*),
AVG(Salary), SUM(Salary)
FROM FACULTY_T
GROUP BY Department_ID
ORDER BY Department_ID DESC;
```

وتكون نتيجة التعليمة السابقة، كما يلي:

DEPART	COUNT(*)	AVG(SALARY)	SUM(SALARY)
STAT	3	39000	117000
PHYS	3	35000	105000
MATH	2	30450	60900
ENGL	3	36833.3333	110500
EE	3	41133.3333	123400
CS	4	35875	143500
CHEM	2	39250	78500

أمَّا إذا أردنا معرفة عدد المواد الدراسية المسجَّل فيها كلُّ طالب في كلِّ فصل دراسي، وحسب السنة الدراسية، وترتيب النتيجة تصاعدياً حسب الأرقام الدراسية للطلبة؛ نستخدم التعليمة التالية:

```
SELECT Student_ID, Year, Semester,
COUNT(*)
FROM ENROLLMENT_T
GROUP BY Student_ID, Year, Semester
ORDER BY Student_ID;
```

ويُلاحَظ في النتيجة التالية للتعليمة أنَّ الطالب رقم 19992020 قد قام بتسجيل أربع مواد دراسية في فصل الخريف من عام 2000م، وأربع مواد دراسية في فصل الربيع من عام 2000م أيضاً، على حين قام الطالب رقم 19992341 بتسجيل ثلاث مواد في فصل الخريف من عام 2000م، ولم يُسجِّل أية مادة أخرى ضِمن أيِّ فصل دراسي آخر.

STUDENT_	YEAR	SEMESTER	COUNT(*)
19992020	2000	FALL	4
19992020	2000	SPRING	4
19992341	2000	FALL	3
19994512	2000	FALL	3
20001111	2000	FALL	4
20001111	2000	SPRING	4
20001212	2000	FALL	1

وفي حالة غياب قِيَم أحد حقول التجميع؛ فإنه يتمُّ إنشاء مجموعة مستقلة للسجلات التي تكون قيم حقول التجميع فيها غائبة.

10-1-2-7 عبارة ترشيح المجموعات الفرعية (HAVING):

تُستخدَم عبارة (HAVING) بشكلٍ اختياريٍّ عند وجود عبارة «تجميع حسب»؛ وذلك لترشيح المجموعات الفرعية؛ بحيث يجب أن تتحقق شروط عبارة (HAVING) على المجموعات الفرعية حتى يمكن أن تظهر ضِمن نتيجة تعليمة الاختيار. ويعني هذا أن عبارة (HAVING) تحتوي على شروط يجب أن تتحقق على المجموعات الفرعية الناتجة من عبارة «التجميع حسب»؛

حتى يمكن أن تظهر ضِمن تعليمة الاختيار. ويمكن تشبيه عبارة (HAVING) بعبارة (WHERE)؛ إذ إن كلتيهما تمثلان شروطاً لعملية الاختيار؛ غير أن عبارة (WHERE) تحتوي على شروط لاختيار السجلات من الجدول، في حين أن عبارة (HAVING) تحتوي على شروط لاختيار المجموعات الفرعية.

فعلى سبيل المثال: إذا أردنا معرفة أرقام الطلاب الذين قاموا بالتسجيل في مواد دراسية يقل عددها عن أربع مواد في أيِّ فصل دراسي من أيِّ عام دراسي، وترتيب النتيجة تصاعدياً حسب الأرقام الدراسية للطلبة، نستخدم التعليمة التالية:

```
SELECT Student_ID, Year, Semester,
COUNT(*)
FROM ENROLLMENT_T
GROUP BY Student_ID, Year, Semester
HAVING COUNT(*) < 4
ORDER BY Student_ID;
```

وتكون نتيجة التعليمة السابقة كما يلي:

```
STUDENT_        YEAR SEMESTER     COUNT(*)
-------- ---------- ---------- ----------
19992341        2000 FALL               3
19994512        2000 FALL               3
20001212        2000 FALL               1
```

ويُلاحَظ في النتيجة السابقة أنه قد تمَّ تقسيم سجلات الجدول إلى مجموعات حسب رمز الطالب، والسنة الدراسية، والفصل الدراسي. بعد ذلك تمَّ حساب عدد المواد الدراسية التي تمَّ التسجيل فيها حسب هذا التقسيم وتطبيق شروط عبارة (HAVING) على المجموعات الفرعية، ومن ثم إظهار المجموعات الفرعية التي تنطبق عليها شروط عبارة (HAVING).

ويمكِن تصوُّر أولويات تنفيذ التعليمة السابقة من قِبل نظام إدارة قاعدة البيانات، كما يلي:

1- اختيار الجدول المناسب حسب ذكره في عبارة مصدر الاختيار (FROM).

2- اختيار سجلات الجدول التي تحقق شروط عبارة شرط الاختيار (WHERE).

3- تقسيم الصفوف التي تحقق شرط الاختيار إلى مجموعات فرعية حسب حقول التقسيم في عبارة (GROUP BY).

4- حذف المجموعات التي لا تحقق الشروط الواردة في عبارة (HAVING).

5- تنفيذ دوال التجميع والتعبيرات الحسابية الواردة في عبارة الاختيار (SELECT) على المجموعات الفرعية التي حققت شروط عبارة (HAVING) أعلاه.

6- ترتيب نتائج عبارة الاختيار (SELECT) الناتجة من الخطوة السابقة حسب عبارة الترتيب (ORDER BY).

7-2-1-11 استخدام تعليمات المجموعات لدمج نتائج تعليمات اختيار متعددة:

حيث إن نتيجة أيِّ عملية اختيار عبارة عن مجموعة متعددة (Multiset or Bag)؛ لكونها قد تحتوي على سجلات متكررة على خلاف المجموعات التي لا تكرَّر فيها العناصر؛ فإنه من الطبيعي أن توفر لغة الاستفسار البنائية ثلاث تعليمات للتعامل مع نتائج الاستفسارات على أنها مجموعات. وهذه التعليمات، هي: الاتحاد (UNION)، والتقاطع (INTERSECT)، والفرق (MINUS).

7-2-1-11-1 الاتحاد (UNION):

تُستخدَم تعليمة الاتحاد (UNION)؛ لدمج نتائج تعليمات اختيار متعددة في جدول نتائج واحد. وعلى الرغم من أن نتائج عمليات الاختيار قد تكون من جداول مختلفة؛ فإنَّ هذه النتائج يجب أن تكون متوافقةً من حيث عملية الاتحاد؛ بمعنى أنها يجب أن تكون بالعدد نفسه من الحقول ومن النوعية نفسها من البيانات (كما سبق أن أوضحنا في الجزء 4-2-2-1-1). ومن القواعد المتبعة في عملية الاتحاد، ما يلي:

1- تُنفذ تعليمات الاختيار بالتسلسل.

2- يجب أن تحتوي جميع عمليات الاختيار التي تربطها عملية الاتحاد على العدد نفسه من الحقول، ومن النوعية نفسها من البيانات.

3- تكون نتيجة عملية الاتحاد جدولاً واحداً لا تكرَّر فيه قِيم السجلات.

4- يُمكِن ترتيب نتيجة عملية الاتحاد باستخدام عبارة الترتيب (ORDER BY) التي تكون بعد آخر عملية اختيار مع استخدام الأرقام النسبية للحقول التي سيتمُّ ترتيب النتائج وفقاً لها (في حالة عدم توافق مسمياتها).

5- عند الرغبة في إظهار السجلات المتكررة؛ يُمكِن استخدام عبارة (UNION ALL) عوضاً عن عبارة (UNION) فقط.

فعلى سبيل المثال: لنفترض أننا نرغب في معرفة أرقام أعضاء هيئة التدريس المؤهلين لتدريس أكثر من مادتين دراسيتين، أو أعضاء هيئة التدريس الذين تقل رواتبهم أو تساوي 000.35 ألفاً، وترتيب النتيجة تصاعدياً وفق أرقام أعضاء هيئة التدريس. في مثل هذه الحالة يُمكِن استخدام تعليمة الاتحاد التالية التي تربط بين نتائج تعليمتي اختيار:

```
SELECT FaCulty_ID
FROM
QUALIFICATION_T
GROUP BY FaCulty_ID
HAVING COUNT(*) > 2
UNION
SELECT FaCulty_ID
FROM FACULTY_T
WHERE Salary <= 35000
ORDER BY FaCulty_ID;
```

وتكون نتيجة تعليمة الاختيار الأولى التي تظهر أرقام هيئة التدريس المؤهلين لتدريس أكثر من مادتين دراسيتين، كما يلي:

```
FACULTY_
--------
200
220
```

أما نتيجة تعليمة الاختيار الثانية التي تظهر أرقام هيئة التدريس الذين تقل رواتبهم أو تساوي 35 ألفاً فهي، كما يلي:

```
FACULTY_
--------
200
220
310
340
400
560
600
710
730
850
```

ونظراً لكون نتائج كلتا عمليتي الاختيار متوافقةً من حيث الاتحاد - تحتويان على العدد نفسه من الحقول ومن النوعية نفسها من البيانات - فإنه يمكِن استخدام عملية الاتحاد بينهما، ومن ثم ترتيب النتيجة تصاعدياً وفقاً لأرقام أعضاء هيئة التدريس، كما توضِّح النتيجة النهائية للتعليمة بشكلها النهائي الذي تمَّ فيه إلغاء الصفوف المتكررة من النتيجة النهائية (وإظهارها مرةً واحدة فقط)، كما يلي:

```
FACULTY_
--------
200
220
310
340
400
560
600
710
730
850
```

أمَّا إذا أردنا إظهار السجلات في حال تكرارها ضِمن نتيجتَيْ عمليتَيْ الاختيار؛ فإنه يمكِن استخدام تعليمة الاتحاد وفق الصيغة التالية:

```
SELECT FaCulty_ID
FROM
QUALIFICATION_T
GROUP BY FaCulty_ID
HAVING COUNT(*) > 2
UNION ALL
SELECT FaCulty_ID
FROM FACULTY_T
WHERE Salary <= 35000
ORDER BY FaCulty_ID;
```

وتكون نتيجة التعليمة التي يظهر فيها رقم عضو هيئة التدريس (200) ورقم عضو هيئة التدريس (220) بشكلٍ متكرِّر (لكونهما مؤهلين لتدريس أكثر من مادتين دراسيتين وتقل رواتبهما أو تساوي 000.35)، كما يلي:

```
FACULTY_
--------
200
200
220
220
310
340
400
560
600
710
730
850
```

7-2-1-11-2 التقاطع (INTERSECT):

تُستخدَم تعليمة التقاطع (INTERSECT)؛ لإظهار السِّجلات المشتركة الناتجة من تعليمات اختيار متعددة في جدول نتائج واحد. وعلى الرَّغم من أن نتائج عمليات الاختيار قد تكون من جداول مختلفة؛ فإن هذه النتائج يجب أن تكون متوافقةً من حيث عملية الاتحاد (كما سبق أن أوضحنا في الجزء 4-2-2-1-1)، كما هو الحال بالنسبة لتعليمة الاتحاد التي سبق شرحها أعلاه. ومن القواعد المُتبعة في عملية التقاطع، ما يلي:

1- تُنفَّذ تعليمات الاختيار بالتسلسل.

2- يجب أن تحتوي جميع عمليات الاختيار التي تربطها عملية التقاطع على العدد نفسه من الحقول، ومن النوعية نفسها من البيانات.

3- تكون نتيجة عملية التقاطع جدولاً واحداً لا تُكرَّر فيه قِيم السجلات.

4- يُمكِن ترتيب نتيجة عملية التقاطع باستخدام عبارة الترتيب (ORDER BY) التي تكون بعد آخر عملية اختيار مع استخدام الأرقام النسبية للحقول التي سيتم ترتيب النتائج وفقاً لها (في حالة عدم توافق مسمياتها).

فعلى سبيل المثال: لنفترض أننا نرغب في معرفة أرقام أعضاء هيئة التدريس المؤهلين لتدريس أكثر من مادتين دراسيتين وتقل رواتبهم أو تساوي 35 ألفاً، وترتيب النتيجة تصاعدياً وفق

أرقام أعضاء هيئة التدريس. في مثل هذه الحالة يمكن استخدام تعليمة التقاطع التالية التي تربط بين نتائج تعليمتَيْ اختيار:

```
SELECT FaCulty_ID
FROM
QUALIFICATION_T
GROUP BY FaCulty_ID
HAVING COUNT(*) > 2
INTERSECT
SELECT FaCulty_ID
FROM FACULTY_T
WHERE Salary <= 35000
ORDER BY FaCulty_ID;
```

وتتكون نتيجة تعليمة التقاطع السابقة من رقمين يمثلان أرقام أعضاء هيئة التدريس المؤهلين لتدريس أكثر من مادتين دراسيتين، ويقل راتب كلٍّ منهم أو يساوي 35.000، كما يلي:

```
FACULTY_
--------
200
220
```

7-2-1-11-3 الفرق (MINUS):

تُستخدَم تعليمة الفرق (MINUS)؛ لإظهار السجلات الناتجة من تعليمة الاختيار الأولى وغير موجودة في أيٍّ من تعليمات الاختيار اللاحقة. وعلى الرَّغم من أن نتائج عمليات الاختيار قد تكون من جداول مختلفة؛ فإن هذه النتائج يجب أن تكون متوافقة من حيث عملية الاتحاد، كما هو الحال بالنسبة لتعليمة الاتحاد وتعليمة التقاطع اللتين سبق شرحهما أعلاه. كما تجدر الإشارة إلى أن لغة الاستفسار البنائية تستخدم مُسمَّى «عدا» (EXCEPT) عوضاً عن مُسمَّى «الفرق»

(MINUS)، إلا أن غالبية النظم التجارية ما زالت تستخدم مُسمَّى الفرق. ومن القواعد المتبعة في عملية الفرق، ما يلي:

1- تُنفَّذ تعليمات الاختيار بالتسلسل.

2- يجب أن تحتوي جميع عمليات الاختيار التي تربطها عملية الفرق على العدد نفسه من الحقول ومن النوعية نفسها من البيانات.

3- تكون نتيجة عملية الفرق جدولاً واحداً لا تُكرَّر فيه قِيم السجلات.

4- يُمكِن ترتيب نتيجة عملية الفرق باستخدام عبارة الترتيب (ORDER BY) التي تكون بعد آخر عملية اختيار مع استخدام الأرقام النسبية للحقول التي سيتم ترتيب النتائج وفقاً لها (في حالة عدم توافق مسمياتها).

فعلى سبيل المثال: لنفترض أننا نرغب في معرفة أرقام أعضاء هيئة التدريس الذين يتقاضون أكثر من 35 ألفاً ومؤهلين لتدريس مواد دراسية يقلُّ عددها عن مادتين دراسيتين، وترتيب النتيجة تصاعدياً وفق أرقام أعضاء هيئة التدريس. في مثل هذه الحالة يمكِن استخدام تعليمة الفرق التالية التي تربط بين نتائج تعليمتَيْ اختيار:

```
SELECT FaCulty_ID
FROM FACULTY_T
WHERE Salary > 35000
MINUS
SELECT FaCulty_ID
FROM QUALIFICATION_T
GROUP BY FaCulty_ID
HAVING COUNT(*) >= 2
ORDER BY FaCulty_ID;
```

وتكون نتيجة تعليمة الاختيار الأولى التي تظهر أرقام هيئة التدريس الذين يتقاضون أكثر من 35 ألفاً، كما يلي:

```
FACULTY_
--------
320
330
420
500
540
640
660
770
800
810
```

أمَّا نتيجة تعليمة الاختيار الثانية؛ فتظهر أرقام أعضاء هيئة التدريس المؤهلين لتدريس مادتين دراسيتين فأكثر فهي، كما يلي:

```
FACULTY_
--------
200
220
320
810
```

ونظراً لكون نتائج كلتا عمليتَيْ الاختيار السابقتين متوافقتين من حيث الاتحاد ـ تحتويان على العدد نفسه من الحقول ومن النوعية نفسها من البيانات ـ فإنه يمكن استخدام عملية الفرق بينهما، ومن ثم ترتيب النتيجة تصاعدياً وفقاً لأرقام أعضاء هيئة التدريس، كما توضح النتيجة النهائية للتعليمة وهي، كما يلي:

```
FACULTY_
--------
330
420
500
540
640
660
770
800
```

وتتكون نتيجة تعليمة الفرق السابقة من ثمانية أرقام تمثل أرقام أعضاء هيئة التدريس الذين يتقاضون أكثر من 35 ألفاً ومؤهلين لتدريس مواد دراسية يقل عددها عن مادتين دراسيتين.

الفصل الثامن

لغة الاستفسار البنائية – الجزء الثاني

يستكمل هذا الفصلُ شرح مكوِّنات لغة الاستفسار البنائية؛ بحيث خُصِّص الجزء الأول منه لاستكمال شرح تعليمة الاختيار، عندما تقوم التعليمة بالتعامل مع أكثر من جدول في آنٍ واحد، ولشرح بقية تعليمات لغة معالجة البيانات؛ وهي: الإضافة، والحذف، والتحديث. أما الجزء الثاني؛ فقد خُصِّص لشرح تعليمات لغة التحكُّم في البيانات.

8-1 الضرب الكرتيزي ورَبط الجداول في تعليمة الاختيار (**Cartesian ProduCt and joining Tables**):

إنَّ عدم وجود عبارة شرط الاسترجاع (Where) ضِمن تعليمة الاختيار (SeleCt) يعني عدم وجود أية شروط على نتيجة التعليمة. ويعني هذا أن كافة سجلات الجدول المُدرَج في عبارة مصدر الاسترجاع (From) مؤهلة لتكون ضِمن نتيجة التعليمة (كما أوضحنا في الفصل السابق عند شرح تعليمة الاختيار). وعندما يتمُّ تحديد أكثر من جدول ضِمن عبارة مصدر الاسترجاع دون وجود عبارة شرط الاسترجاع؛ فإنَّ نتيجة التعليمة تكون الضرب الكرتيزي لسجلات الجداول المدوَّنة في عبارة مصدر الاسترجاع. ويعني هذا أن كافة التوليفات بين سجلات الجداول ستكون ضِمن نتيجة تعليمة الاختيار. فعلى سبيل المثال: لنفترض وجود جدول أعضاء هيئة التدريس (FaCulty_T) وجدول المجموعات الدراسية (SeCtion_T) ضِمن عبارة مصدر الاسترجاع، كما يلي:

```
SELECT *
```

```
FROM FACULTY_T,
SECTION_T;
```

سيكون عدد السجلات المُسترجَعة عند تنفيذ التعليمة السابقة أربعمائة سجل؛ وذلك لكون كلّ سجل من سجلات أعضاء هيئة التدريس (وعددها عشرون) سيرتبط بـكلِّ سجلٍ من سجلات المجموعات الدراسية (وعددها عشرون أيضاً). وعلى الرغم من أن التعليمة السابقة بشكلها السابق قلَّما تُستخدَم على أرض الواقع؛ لكون نتيجتها ليست ذات معنى، غير أنها تُعَدُّ أساساً لعمليات الرَّبط بين الجداول في لغة الاستفسار البـنائية. وعمليات الربط هي الوسيلة الوحيدة التي تمكِّننا من الانتقال من جدولٍ إلى آخر ورَبط البـيانات الموجودة في سجلات الجداول المختلفة. وتتمُّ عملية الربط من خلال استخدام عوامل المقارنة بين حقول الجداول قيد الربـط. فعلى سبيل المثال: لمعرفة أسماء الطلبة وأرقام المواد الدراسية التي سجَّلوا فيها ونتائجهم في هذه المواد؛ يتمُّ ربط جدول التسجيل (ENROLLMENT_T) مع جدول الطلبة (STUDENT_T)، كما يلي:

```
SELECT FName, LName, Course_ID, Grade
FROM STUDENT_T, ENROLLMENT_T
WHERE STUDENT_T.Student_ID =
ENROLLMENT_T.Student_ID;
```

ويُمكِن فهم طريقة عمل التعليمة السابقة، كما يلي:

1- إجراء عملية الضرب الكرتيزي بين جدول المواد المُسجَّلة (ENROLLMENT_T)، وجدول الطلبة (STUDENT_T). ويكون ناتج هذه العملية كافة توليفات السجلات الموجودة في جدول المواد المُسجَّلة وجدول الطلـبة. وتكون حقول السجلات الناتجة من هذه العملية عبارة عن كافة حقول الجدول الأول متبوعةً بكافة حقول الجدول الثاني.

2- يُطبق عامل المقارنة «يساوي» (=) على ناتج عملية الضرب الكرتيزي؛ بحيث تكون قيمة الحقل «رقم الطالب» الذي تمَّ جلبه من جدول الطلبة (والذي يمثل المفتاح الرئيسي لجدول الطلبة) مساوياً لحقل «رقم الطالب» الذي تمَّ جلبـه من جدول المواد المسجَّلة (الذي يمثل مفتاحاً

خارجياً في جدول المواد المسجلة، وفي نفس الوقت، جزءاً من المفتاح الرئيسي للجدول). وعند تساوي هذين الحقلين لسجل ما في ناتج عملية الضرب الكرتيزي؛ يكون هذا السجل أحد السجلات الناتجة من عملية الاختيار.

3- بعد معرفة السجلات الناتجة من عملية الربط، أيْ: العمليتين السابقتين؛ يتم اختيار الحقول المطلوبة (وعددها أربعة حقول) حسب ورودها في تعليمة الاختيار.

وبناءً على ذلك، يكون ناتج التعليمة السابقة، كما يلي:

FNAME	LNAME	COURSE_	GRADE
Saleh	Alhamad	CHEM101	4
Abdullah	Aloufi	CHEM101	3
Khalid	Alsultan	CHEM101	4
Salem	Algamdi	CHEM101	3
Mishal	Alyousef	CHEM101	1
Saleh	Alhamad	CS101	2
Mishal	Alyousef	CS101	4
Saleh	Alhamad	CS102	3
Mishal	Alyousef	CS102	4
Saleh	Alhamad	ENGL101	3
Abdullah	Aloufi	ENGL101	4
Salem	Algamdi	ENGL101	4
Mishal	Alyousef	ENGL101	4
Saleh	Alhamad	ENGL102	1
Mishal	Alyousef	ENGL102	4
Saleh	Alhamad	MATH101	3
Abdullah	Aloufi	MATH101	2
Salem	Algamdi	MATH101	0
Mishal	Alyousef	MATH101	2
Saleh	Alhamad	MATH102	2
Mishal	Alyousef	MATH102	0
Saleh	Alhamad	STAT101	2
Mishal	Alyousef	STAT101	3

كما يُمكِن وَضْع أية شروط على ناتج عملية الرَّبط، أو ترتيب نتائجها وفقاً لترتيب معين. فعلى سبيل المثال: لمعرفة أسماء الطلبة الذين درسوا في مواد الحاسب الآلي أو مواد الرياضيات، ودرجاتهم، وترتيب النتيجة تصاعدياً حسب الاسم الأول للطلبة؛ يمكِن استخدام التعليمة التالية:

```
SELECT FName, LName, Course_ID, Grade
FROM STUDENT_T, ENROLLMENT_T
```

```
WHERE STUDENT_T.Student_ID = ENROLLMENT_T.Student_ID
AND (Course_ID LIKE 'CS%' OR Course_ID LIKE 'MATH%')
ORDER BY FName ASC;
```

وتكون نتيجة التعليمة السابقة، كما يلي:

FNAME	LNAME	COURSE_	GRADE
Abdullah	Aloufi	MATH101	2
Mishal	Alyousef	MATH101	2
Mishal	Alyousef	CS101	4
Mishal	Alyousef	CS102	4
Mishal	Alyousef	MATH102	0
Saleh	Alhamad	MATH101	3
Saleh	Alhamad	CS102	3
Saleh	Alhamad	CS101	2
Saleh	Alhamad	MATH102	2
Salem	Algamdi	MATH101	0

وفي هذه الحالة يمكِن فهم طريقة عمل التعليمة السابقة؛ بالإضافة لما سبق أعلاه في (1) و (2)، كما يلي:

3- بعد تحديد السجلات المؤهلة من عملية الربط؛ تُطبق عليها الشروط الواردة في عبارة شرط الاختيار.

4- يتمُّ اختيار الحقول المطلوبة (وعددها أربعة حقول) حسب ورودها في تعليمة الاختيار.

5- يتمُّ ترتيب سجلات النتيجة حسب الحقول الواردة في عبارة «ترتيب حسب» (Order By) وحسب ترتيب الحقول في العبارة (في حالة الترتيب وفق أكثر من حقل).

وتُسمَّى عملية الربط التي يُستخدَم فيها عامل المساواة (=) بعملية «ربط التساوي» (Equi-Join). غير أنه يُمكِن استخدام العوامل الأخرى، وهي: (<>, <, >, =<, =>). وتمثِّل العملية في هذه الحالة ما يُعرَف بالربط العام (Theta-Join). ونظراً لأهمية عملية الربط بين الجداول المختلفة للحصول على بيانات مترابطة منطقياً فيما بينها في النموذج العلاقي، توفر لغة الاستفسار البنائية عدداً من عبارات الربط، بالإضافة إلى استخدام الربط بالطريقة السابقة. ومن هذه العبارات عبارة «الربط» (JOIN) التي يمكن استخدامها لتمثيل عملية الربط السابقة، كما يلي:

```
SELECT FName, LName, Course_ID, Grade
FROM STUDENT_T JOIN ENROLLMENT_T
ON STUDENT_T.Student_ID =
ENROLLMENT_T.Student_ID
WHERE Course_ID LIKE 'CS%' OR Course_ID LIKE
'MATH%'
ORDER BY FName ASC;
```

ففي المثال السابق؛ تمَّ استخدامُ عبارة الرَّبط ضِمن عبارة مصدر الاختيار، وتم تحديد الحقل الذي ستطبق من خلاله عملية الرَّبط بعد كلمة (ON). وتُعَدُّ هذه الطريقة أسهل للفهم من الطريقة السابقة؛ لأنها لا تدمج بين شروط الاختيار وعمليات الربط ضِمن عبارة شرط الاختيار؛ ولكنها تفصل بينهما من خلال إدراج عملية الربط في عبارة مصدر الاختيار وإدراج شروط الاختيار ضِمن عبارة شرط الاختيار. وتكون نتيجة العملية السابقة جدولاً واحداً يحتوي على كافة حقول جدول الطلبة متبوعة بكافة حقول جدول المواد الدراسية المسجَّلة. كما يُلاحَظ ذكر اسم الجدول قبل اسم الحقل الذي ستُطبق من خلاله عملية الربط؛ وذلك لإزالة الالتباس بين مُسمَّيات الحقول خاصة، وأن اسم هذا الحقل متكرر في الجدولين. ويمكن اختصار أسماء الجدولين؛ لتقليص طول عملية الاختيار من خلال عملية إعادة تسميتهما، كما يلي:

```
SELECT FName, LName, Course_ID, Grade
FROM STUDENT_T S JOIN ENROLLMENT_T E
        ON S.Student_ID = E.Student_ID
WHERE Course_ID LIKE 'CS%' OR Course_ID LIKE
'MATH%'
ORDER BY FName ASC;
```

وتوفر لغة الاستفسار البنائية أنواعاً مختلفة من الربط من ضِمنها «الربط الطبيعي» (Natural Join) وأنواعاً من «الربط الخارجي» (Outer Join). ولإيضاح فكرة الربط

الطبيعي، لنفترض أننا نرغب في إظهار كافة الحقول الناتجة بعد عملية ربـط جدول المواد الدراسية مع جدول الأقسام الدراسية. في هذه الحالة، تستخدم تعليمة الاختيار التالية:

```
* SELECT
FROM COURSE_T C JOIN DEPARTMENT_T D
ON C.Department_ID = D.Department_ID;
```

وتكون نتيجة التعليمة السابقة؛ هي ربط بيانات كلِّ مادة دراسية ببيانات القسم الدراسي الذي يقدِّمها، كما يلي:

COURSE_	TITLE	UNITS	DEPART	DEPART	NAME
CHEM101	CHEMISTRY (I)	3	CHEM	CHEM	Chemistry
CHEM102	CHEMISTRY (II)	3	CHEM	CHEM	Chemistry
CS101	JAVA PROGRAMMING	3	CS	CS	Computer Science
CS102	SOFTWARE ENGINEERING	3	CS	CS	Computer Science
CS103	C/C++ PROGRAMMING	3	CS	CS	Computer Science
CS104	COMPUTER ARCHITECTURE	3	CS	CS	Computer Science
CS105	INTRODUCTION TO DATABASE SYSTEMS	3	CS	CS	Computer Science
EE101	ELECTRIC CIRCUITS	3	EE	EE	Electrical Engineering
EE102	ELECTRONICS (I)	3	EE	EE	Electrical Engineering
EE103	ELECTRONICS (II)	3	EE	EE	Electrical Engineering
EE104	COMMUNICATION NETWORKS	4	EE	EE	Electrical Engineering
ENGL101	ENGLISH GRAMMAR	2	ENGL	ENGL	English Language
ENGL102	ENGLISH WRITING	3	ENGL	ENGL	English Language
ENGL103	TECHNICAL WRITING	3	ENGL	ENGL	English Language
MATH101	INTRODUCTION To MATHEMATICS	3	MATH	MATH	Mathematics
MATH102	DIFFERENTIAL EQUATIONS	3	MATH	MATH	Mathematics
MATH103	CALCULUS (I)	3	MATH	MATH	Mathematics
MATH104	CALCULUS (II)	3	MATH	MATH	Mathematics
MATH106	ALGEBRA	4	MATH	MATH	Mathematics
MATH107	COMPUTER MATHEMATICS	3	MATH	MATH	Mathematics
PHYS101	PHYSICS (I)	3	PHYS	PHYS	Physics
PHYS102	PHYSICS (II)	3	PHYS	PHYS	Physics
STAT101	INTRODUCTION TO STATISTICS	3	STAT	STAT	Statistics
STAT102	ADVANCED STATISTICS	3	STAT	STAT	Statistics

ويُلاحَظ في النتيجة السابـقة أنَّ حقل رمز القسم الدراسي (الذي يمثل المفتاح الرئيسي لجدول الأقسام الدراسية، والمفتاح الخارجي لجدول المواد الدراسية) قد تكرَّر مرتين. ونظراً لأن تكرار بيانات هذا الحقل لا تضيف أية معلومة جديدة للنتيجة؛ فإنه يمكن إلغاء أحدهما دون الإخلال بـالنتيجة. ويمكن إلغاء المتكرِّر من حقول الربط باستخدام «الربط الطبيعي» الذي يقوم بالربط بين جدولين، ضمنياً، حسب الأسماء المتطابقة للحقول فيهما (والتي قد تكون أكثر من زوج). فعلى سبيل المثال: يمكِن تنفيذ الاستفسار السابق باستخدام الربط الطبـيعي، كما يلي:

```
SELECT *
FROM COURSE_T NATURAL JOIN DEPARTMENT_T;
```

وتكون نتيجة العملية السابقة مكوَّنة من خمسة حقول، عوضاً عن ستة حقول؛ بحيث تمَّ إلغاء المتكرر وهو حقل رمز القسم الدراسي وإظهاره مرةً واحدة فقط ضِمن نتيجة العملية، وكأول حقل في جدول النتيجة، كما يلي:

DEPART	COURSE_	TITLE	UNITS	NAME
CHEM	CHEM101	CHEMISTRY (I)	3	Chemistry
CHEM	CHEM102	CHEMISTRY (II)	3	Chemistry
CS	CS101	JAVA PROGRAMMING	3	Computer Science
CS	CS102	SOFTWARE ENGINEERING	3	Computer Science
CS	CS103	C/C++ PROGRAMMING	3	Computer Science
CS	CS104	COMPUTER ARCHITECTURE	3	Computer Science
CS	CS105	INTRODUCTION TO DATABASE SYSTEMS	3	Computer Science
EE	EE101	ELECTRIC CIRCUITS	3	Electrical Engineering
EE	EE102	ELECTRONICS (I)	3	Electrical Engineering
EE	EE103	ELECTRONICS (II)	3	Electrical Engineering
EE	EE104	COMMUNICATION NETWORKS	4	Electrical Engineering
ENGL	ENGL101	ENGLISH GRAMMAR	2	English Language
ENGL	ENGL102	ENGLISH WRITING	3	English Language
ENGL	ENGL103	TECHNICAL WRITING	3	English Language
MATH	MATH101	INTRODUCTION To MATHEMATICS	3	Mathematics
MATH	MATH102	DIFFERENTIAL EQUATIONS	3	Mathematics
MATH	MATH103	CALCULUS (I)	3	Mathematics
MATH	MATH104	CALCULUS (II)	3	Mathematics
MATH	MATH106	ALGEBRA	4	Mathematics
MATH	MATH107	COMPUTER MATHEMATICS	3	Mathematics
PHYS	PHYS101	PHYSICS (I)	3	Physics
PHYS	PHYS102	PHYSICS (II)	3	Physics
STAT	STAT101	INTRODUCTION TO STATISTICS	3	Statistics
STAT	STAT102	ADVANCED STATISTICS	3	Statistics

إن عملية الرَّبط الافتراضية هي «الربط الداخلي» (Inner Join)؛ بمعنى أنه يتمُّ إضافة سجل لنتيجة عملية الربط إذا توفر سجل في أحد الجدولين وسجل مكافئ له في الجدول الآخر وفق حقل (أو حقول) الربـط. وبالنظر في نتيجة المثال الذي يتطرق لمعرفة أسماء الطلبة وأرقام المواد الدراسية التي سجلوا فيها ونتائجهم في هذه المواد، أعلاه؛ فإن النتيجة تحتوي على بعض الطلبـة (وليس جميعهم). وهؤلاء الطلبـة هم الذين يُوجَد لهم سجلات في جدول المواد المسجلة. أما إذا أردنا معرفة أسماء الطلبة وأرقام المواد الدراسية التي سجلوا فيها ونتائجهم في هذه المواد بالإضافة إلى الطلبة الذين لم يقوموا بتسجيل مواد دراسية؛ فإننا نستخدم «الربط الخارجي الأيسر» (Left Outer Join). وحتى نقلِّص حجم نتيجة العملية، سنشترط كون الطالب تابعاً لتخصص الحاسب الآلي. في هذه الحالة تكون تعليمة الاختيار، كما يلي:

```
SELECT FName, LName, Course_ID, Grade
FROM STUDENT_T S LEFT OUTER JOIN
ENROLLMENT_T E
        ON S.Student_ID = E.Student_ID
WHERE S.Major = 'CS';
```

ويُلاحَظ في النتيجة التالية للتعليمة ظهور كافة طلبة الحاسب الآلي، وظهور نتيجة المواد الدراسية للمسجّلين منهم في مواد دراسية. أمَّا بالنسبة لغير المسجلين (وعددهم واحد فقط)؛ فقد تمَّ إدراجُ قيم غير معروفة ضِمن حقول المواد المسجلين فيها. ويعني هذا أن نتيجة عملية «الربط الخارجي الأيسر» ستخرج ضِمن نتائجها كافة الحقول المحدَّدة في تعليمة الاختيار لكافة سجلات الجدول الأيسر للعملية (وهو جدول الطلبة) بغض النظر عن وجود ما يطابقها في الجدول الأيمن.

FNAME	LNAME	COURSE_	GRADE
Saleh	Alhamad	CHEM101	4
Mishal	Alyousef	CHEM101	1
Saleh	Alhamad	CS101	2
Mishal	Alyousef	CS101	4
Saleh	Alhamad	CS102	3
Mishal	Alyousef	CS102	4
Saleh	Alhamad	ENGL101	3
Mishal	Alyousef	ENGL101	4
Saleh	Alhamad	ENGL102	1
Mishal	Alyousef	ENGL102	4
Saleh	Alhamad	MATH101	3
Mishal	Alyousef	MATH101	2
Saleh	Alhamad	MATH102	2
Mishal	Alyousef	MATH102	0
Saleh	Alhamad	STAT101	2
Mishal	Alyousef	STAT101	3
Ghanim	Alhmoud		

أمَّا عملية «الرَّبط الخارجي الأيمن» (Right Outer Join)؛ فتعمل عكس عملية «الربط الخارجي الأيسر». ويعني هذا إظهار كافة الحقول المُحدَّدة في تعليمة الاختيار لكافة سجلات الجدول الأيمن بغض النظر عن وجود ما يطابقها في الجدول الأيسر من سجلات. فعلى سبيل المثال: لمعرفة المواد التابعة لقسم الحاسب الآلي ("CS") التي تمَّ تسجيل بعض الطلبة فيها، وكذلك المواد

الدراسية التابعة لقسم الحاسب الآلي التي لم يُسجَّل فيها أية طالب مع إظهار أرقام الطلبة المسجَّلين في المواد التي تمَّ التسجيل فيها، يمكِن استخدام «الربط الخارجي الأيمن»، كما يلي:

```
SELECT Student_ID, C.Course_ID, C.Title
FROM ENROLLMENT_T E RIGHT OUTER JOIN
COURSE_T C
Course_ID = C.Course_ID      ON E.
WHERE C.Course_ID LIKE 'CS%';
```

ويُلاحَظ في النتيجة التالية للتعليمة ظهورُ كافة مواد الحاسب الآلي، وظهور أرقام الطلبة للمواد التي تمَّ التسجيل فيها. أمَّا بالنسبة للمواد التي لم يُسجَّل فيها أيُّ طالب؛ فتظهر المادة الدراسية دونما قِيم مدوَّنة في حقل رقم الطالب.

```
STUDENT_ COURSE_ TITLE
-------- ------- ----------------------------------
19992020 CS101   JAVA PROGRAMMING
20001111 CS101   JAVA PROGRAMMING
19992020 CS102   SOFTWARE ENGINEERING
20001111 CS102   SOFTWARE ENGINEERING
         CS103   C/C++ PROGRAMMING
         CS104   COMPUTER ARCHITECTURE
         CS105   INTRODUCTION TO DATABASE SYSTEMS
```

بالإضافة للرَّبط الخارجي الأيمن والربط الخارجي الأيسر؛ توفِّر لغة الاستفسار البنائية «الربط الخارجي الكامل» (Full Outer Join). وعند استخدام الربط الخارجي الكامل تكون النتيجة إظهار حقول السجلات التي تتطابق في حقول الربط، وإظهار حقول السجلات في الجدول الأيسر التي لا يُوجَد ما يطابقها في الجدول الأيمن، وإظهار حقول سجلات الجدول الأيمن التي لا يُوجَد ما يطابقها في الجدول الأيسر. ويتمُّ استكمال بقية الحقول في الجدول الناتج بالقيم الغائبة «NULL» لحالات عدم التطابق سواء من قِبل سجلات الجدول الأيمن أو الجدول الأيسر.

كما يُمكِن استخدامُ كلمة «طبيعي» (Natural)؛ للرَّبط بين جدولين، ضِمنياً، من خلال الحقول المتطابقة في الجدولين؛ بالإضافة إلى إزالة حقل (أو حقول) الربط المتكررة من نتيجة تعليمة الاختيار. فعلى سبيل المثال: يُستخدَم «الربط الطبيعي الأيسر» (Natural Left Outer Join) عند الرَّغبة في إجراء الربط الأيسر بين جدولين، وإظهار الحقول المتكرِّرة مرةً واحدة فقط ضِمن النتيجة النهائية للتعليمة. توفر لغة الاستفسار البنائية أيضاً عبارة «الضرب الكرتيزي» (Cross Join) لاستخدامها بشكلٍ ظاهرٍ عوضاً عن استخدام مُسمَّيات الجداول فقط في عبارة مصدر الاسترجاع. وعلى الرغم من أن كافة تعليمات الربط السابقة يمكِن صياغتها من خلال العبارات الأخرى التي توفرها لغة الاستفسار البنائية؛ فإنَّ وجود هذه العبارات ضِمن لغة الاستفسار البنائية يُسهِّل عملية قراءة المقصود منها مقارنة باستخدام تعليمات قد تكون أكثر تعقيداً في فهمها للحصول على نفس النتيجة. بالإضافة إلى ذلك؛ فإن وجود هذه العبارات يسهِّل، وبشكلٍ كبير، كتابة تعليمات الربط المناسبة في لغة الاستفسار البنائية.

من الممكِن أيضاً استخدامُ تعليمات الرَّبط بشكلٍ متداخل، أو ربط أكثر من جدول في نفس الوقت. فعلى سبيل المثال: لمعرفة أعضاء هيئة التدريس الذين يعملون في قسم الحاسب الآلي بالإضافة إلى مُسمَّيات المواد الدراسية المؤهلين لتدريسها، يتوجب ربط جدول أعضاء هيئة التدريس بجدول المواد الدراسية من خلال جدول المؤهلات التدريسية (QulifiCation_T)، كما يلي:

```
SELECT FName, LName, Title Qualified_to_TeaCh
FROM (FACULTY_T NATURAL JOIN QUALIFICATION_T) JOIN
COURSE_T
ON QUALIFICATION_T.Course_ID =
    Course_T.Course_ID
WHERE Department_ID = ‘CS’;
```

وتكون نتيجة التعليمة السابقة، كما يلي:

```
FNAME        LNAME        QUALIFIED_TO_TEACH
------------ ------------ ----------------------------------
Saleh        Aleesa       JAVA PROGRAMMING
Mohammed     Alhamad      SOFTWARE ENGINEERING
Mohammed     Alhamad      C/C++ PROGRAMMING
Ghanim       Alghanim     COMPUTER ARCHITECTURE
Ibraheem     Alsaleh      INTRODUCTION TO DATABASE SYSTEMS
```

أمَّا إذا أردنا معرفة المجموعات الدراسية التابعة لقسم الحاسب الآلي وأسماء الطلبة المُسجَّلين فيها بما في ذلك المجموعات التي لم يُسجَّل فيها أيُّ طالب؛ فيُمكِن استخدام التعليمة التالية:

```
SELECT FName, LName, Course_ID, SeCtion_No, Year, Semester
FROM (((STUDENT_T NATURAL JOIN ENROLLMENT_T)
   NATURAL RIGHT OUTER JOIN SECTION_T)
NATURAL JOIN COURSE_T)
WHERE Department_ID = 'CS';
```

ولقد تمَّ استخدامُ عملية الرَّبط الطبيعي الأولى في التعليمة السابقة؛ لمعرفة الطلبة والمجموعات الدراسية المسجَّلين فيها. أمَّا عملية الربط الثانية؛ فهي ربط طبيعي أيمن يُمكِّننا من معرفة كافة المجموعات الدراسية ومعرفة الطلبة المسجَّلين فيها بما في ذلك المجموعات التي لم يُسجَّل فيها أيُّ طالب. أمَّا عملية الرَّبط الثالثة؛ فتمكِّننا من ربط المجموعات الدراسية والطلبة المسجَّلين فيها بالمواد التي تتبعها هذه المجموعات. وعملية الرَّبط هذه ضرورية؛ لمعرفة القسم الذي تتبعه المجموعات الدراسية، والتي لا تتوفر إلا من خلال جدول المواد الدراسية الذي سيُطبق عليه شرط الاختيار، وهو أن تتبع المجموعة لقسم الحاسب الآلي. كما يُلاحَظ في التعليمة السابقة استخدام الأقواس؛ وذلك لتحديد أولويات عمليات الرَّبط؛ حيث أنه يتمُّ تنفيذ العمليات الواقعة بين الأقواس من الداخل للخارج؛ بمعنى أن العمليات المضمَّنة ضِمن الأقواس الداخلية تتمُّ قبل الأقواس التي تحتويها وهكذا حتى يتمَّ تنفيذ الأقواس الخارجية. وتكون نتيجة التعليمة السابقة، كما يلي:

FNAME	LNAME	COURSE_	SECTION_NO	YEAR	SEMESTER
Saleh	Alhamad	CS101	1	2000	FALL
Mishal	Alyousef	CS101	2	2000	FALL
Saleh	Alhamad	CS102	1	2000	SPRING
Mishal	Alyousef	CS102	1	2000	SPRING
		CS103	1	2000	SPRING
		CS104	1	2001	FALL
		CS105	1	2001	SPRING

كما يُمكِن الاستغناءُ عن عملية الرَّبط الثالثة في التعليمة السابقة والحصول على نفس النتيجة إذا ما لاحظنا أن رمز المادة الدراسية (Course_ID) (ضِمن جدول المجموعات الدراسية) يدلُّ على اسم القسم الدراسي الذي تتبعه المجموعة الدراسية؛ إذ إن رمز المادة الدراسية يتكوَّن من رمز القسم متبوعاً بـرقم المادة الدراسية. وبناءً على ذلك يُمكِن صياغة التعليمة السابقة، كما يلي:

```
SELECT FName, LName, Course_ID, SeCtion_No, Year, Semester
FROM ((STUDENT_T NATURAL JOIN ENROLLMENT_T)
                    NATURAL RIGHT OUTER JOIN
CTION_T)SE
WHERE Course_ID LIKE 'CS%';
```

تُمكِّن لغة الاستفسار البنائية أيضاً من ربـط الجدول مع نفسه من خلال إعطائه مُسمَّيات مختلفة داخل تعليمة الاختيار. وفي هذه الحالة؛ يُمكِن اعتـبار (أو تصوُّر) كلِّ مُسمَّى يُعطى للجدول على أساس أنه نسخة مختلفة للجدول نفسه. فعلى سبيل المثال: لمعرفة المواد الدراسية المؤهَّل لتدريسها أكثر من عضو هيئة تدريس؛ يمكِن استخدامُ التعليمة التالية:

```
SELECT Q1.Course_ID, Q2.FaCulty_ID
FROM (QUALIFICATION_T Q1 JOIN QUALIFICATION_T Q2
ON Q1.Course_ID = Q2.Course_ID)
WHERE Q1.FaCulty_ID <> Q2.FaCulty_ID
GROUP BY Q1.Course_ID, Q2.FaCulty_ID;
```

ففي التعليمة السابقة تمَّت إعادة تسمية جدول المؤهلات التدريسية لأعضاء هيئة التدريس (QUALIFICATION_T) مرتين (مرةً بمُسمَّى Q1 ومرةً بمُسمَّى Q2) للحصول على نسختين (تصوريتين) من جدول المؤهلات التدريسية. وبعد ذلك تمَّ ربط الجدولين من خلال الحقل المشترك «رقم المادة الدراسية» (Course_ID). وحتى يتمَّ التخلُّص من السجلات الناتجة بعد عملية الرَّبط التي يكون فيها رقم عضو هيئة التدريس مكرراً (في حقلين من حقول الجدول الناتج) - تمَّت إضافة شرط عدم التساوي في عبارة شرط الاسترجاع. ومعنى هذا أن السِّجلات التي يتكرَّر فيها نفس رقم عضو هيئة التدريس؛ هي مواد دراسية مؤهل لتدريسها عضو هيئة تدريس واحد، ويتوجَّب حذفها من نتيجة التعليمة. أمَّا بقية السِّجلات؛ فهي سجلات لا يتكرَّر فيها نفس رقم عضو هيئة التدريس؛ مما يعني أنها تُدرَّس من قِبل أساتذة مختلفين. وحتى يتمَّ إظهارُ عضو هيئة التدريس الذي يقوم بتدريس مادة يقوم بتدريسها أعضاء هيئة تدريس آخرين مرةً واحدةً فقط ضِمن نتيجة العملية؛ تمَّ استخدامُ عبارة «التجميع حسب» (Group By)، التي يتوجَّب فيها استخدامُ كافة الحقول المختارة ضِمن تعليمة الاختيار، وهي حقل رقم المادة الدراسية، وحقل رقم عضو هيئة التدريس. وتكون نتيجة التعليمة السابقة، كما يلي:

```
COURSE_ FACULTY_
------- --------
PHYS101 710
PHYS101 770
STAT101 600
STAT101 660
```

أمَّا إذا أردنا معرفة أسماء أعضاء هيئة التدريس، وعدم الاكتفاء بأرقامهم؛ فيمكِن الرَّبط مع جدول أعضاء هيئة التدريس (بالإضافة إلى الرَّبط السابق لجدول المؤهلات التدريسية مع نفسه)، كما يلي:

```
SELECT Q1.Course_ID, F.FName, F.LName, Q2.FaCulty_ID
FROM ((QUALIFICATION_T Q1 JOIN QUALIFICATION_T Q2
Course_ID = Q2.Course_ID) JOIN FACULTY_T F            ON Q1.
ON Q2.FaCulty_ID = F.FaCulty_ID)
WHERE Q1.FaCulty_ID <> Q2.FaCulty_ID
GROUP BY Q1.Course_ID, Q2.FaCulty_ID, F.FName, F.LName;
```

وتكون نتيجة التعليمة السابقة كما يلي:

```
COURSE_ FNAME        LNAME        FACULTY_
------- ------------ ------------ --------
PHYS101 Mahmood      Alsalem      710
PHYS101 Sultan       Aljasir      770
STAT101 Turki        Alturki      600
STAT101 Saud         Alkhalifa    660
```

2-8 الاستفسارات المتداخلة (Nested Queries):

تتطلَّب بعض عمليات الاختيار الحصولَ على قِيم من قاعدة البيانات، ومن ثم استخدام هذه القِيم في عوامل مقارنة ضِمن شرط الاختيار. ويُمكِن تكوين مثل عمليات الاختيار هذه من خلال ما يُعرَف بـالاستفسارات المتداخلة التي يتضمَّن شرط الاسترجاع فيها على عمليات استفسار أخرى. وتُسمَّى عملية الاستفسار الموجودة في عبارة الشرط (Where) بعملية الاستفسار الداخلية (Inner Query)؛ في حين تُسمَّى تعليمة الاستفسار التي تحتويها بعملية الاستفسار الخارجية (Outer Query). فعلى سبيل المثال: لو أردنا معرفة أعضاء هيئة التدريس الذين تزيد رواتبهم عن متوسط رواتب أعضاء هيئة التدريس؛ فإنه يُمكِننا ذلك من خلال تعليمَتَيْ استفسار؛ الأولى للحصول على متوسط رواتب أعضاء هيئة التدريس، كما يلي:

```
SELECT AVG(SALARY)
FROM FACULTY_T;
```

وتكون نتيجة التعليمة السابقة، كما يلي:

```
AVG(SALARY)
-----------
      36940
```

أمَّا التعليمة الثانية؛ فتستخدم النتيجة السابقة؛ لمعرفة أعضاء هيئة التدريس الذين تزيد رواتبهم عن المتوسط الذي سبق حسابه أعلاه، كما يلي:

```
*SELECT
FROM FACULTY_T
WHERE Salary > 36940;
```

وتكون نتيجة التعليمة السابقة، كما يلي:

FACULTY_	FNAME	LNAME	PHONE_NO	SALARY	DOB	DEPART
320	Mohammed	Alhamad	454-5412	44000	13-MAY-65	CS
330	Ghanim	Alghanim	456-2234	44500	12-AUG-69	CS
420	Saleh	Alghamdi	454-2233	44600	13-FEB-69	CHEM
540	Salem	Alhamad	456-3304	40000	11-SEP-72	ENGL
640	Fahad	Alzaid	456-3322	44300	12-MAY-71	STAT
660	Saud	Alkhalifa	454-9856	44900	13-AUG-72	STAT
770	Sultan	Aljasir	456-3212	43300	13-MAY-70	PHYS
800	Ali	Albader	456-7812	45300	22-JUN-66	EE
810	Saad	Alzhrani	454-5578	44200	17-OCT-67	EE

غير أنه باستخدام تعليمات الاستفسار المتداخلة يمكِن دَمْج التعليمتين السابقتين؛ للحصول على نفس النتيجة، ضمن تعليمة استفسار واحدة، كما يلي:

```
*SELECT
FROM FACULTY_T
WHERE Salary > (SELECT AVG(Salary) FROM
FACULTY_T);
```

ويمكِن تصوُّر طريقة عمل الاستفسار المتداخل السابق، كما يلي:

1- تنفذ تعليمة الاختيار الداخلية أولاً.

2- تستخدم نتيجة تعليمة الاختيار الداخلية في تنفيذ عملية الاختيار الخارجية.

ويُعَدُّ الاستفسار الفرعي السابـق من الاستفسارات الفرعية التي تُعيد قيمة واحدة فقط، والتي يُمكِن استخدام أيٍّ من عوامل المقارنة التالية معها: (<>, >, <, =, >=, <=). وعندما يتطلـب الأمر استخدام أكثر من عملية استفسار داخلية؛ يُمكِن ربط الاستفسارات الداخلية من خلال العوامل المنطقية «أو» (OR) «و» (AND). فعلى سبيل المثال: لمعرفة بيانات أعضاء هيئة التدريس الذين تنحصر رواتبهم بين المتوسط العام للرواتب بعد زيادته بنسبة عشرة في المائة، والمعدَّل العام بعد إنقاصه بنسبة عشرة في المائة، يمكِن استخدام الاستفسار المتداخل التالي:

```
SELECT *
  FROM FACULTY_T
  WHERE Salary <= (SELECT AVG(Salary)
  FROM FACULTY_T) * 1.1
   AND Salary >= (SELECT
  AVG(Salary) FROM FACULTY_T) * 0.9;
```

وتكون نتيجة الاستفسار السابق، كما يلي:

```
FACULTY_ FNAME       LNAME       PHONE_NO     SALARY DOB       DEPART
-------- ----------- ----------- -------- ---------- --------- ------
200      Khalid      Aloufi      454-2341      35000 22-MAY-63 MATH
400      Ahmad       Alotaibi    454-4563      33900 17-MAY-71 CHEM
500      Yahya       Khorshid    456-2221      36700 12-MAR-65 ENGL
540      Salem       Alhamad     456-3304      40000 11-SEP-72 ENGL
560      Salman      Albassam    454-7865      33800 13-SEP-68 ENGL
850      Ahmad       Alsabti     456-0120      33900 15-APR-73 EE
```

عند استخدام الاستفسارات المتداخلة؛ يجب توخِّي الدقة في حالة تكرار مُسمَّيات الحقول في الجدول المُستخدَم في الاستفسار الداخلي والجدول المُستخدَم في الجدول الخارجي؛ إذ إنه يُفضَّل دائماً ذِكر اسم الجدول الذي يتبعه الحقل لإزالة الالتبـاس. والحالة الافتراضية في لغة الاستفسار البنائية؛ هي أن أيَّ حقل مُتكرِّر دون ذكر اسم الجدول الذي يتبعه الحقل، سيفترض على أساس أنه الحقل التابع لأكثر الاستفسارات تداخلاً (Innermost Query). ويشابه هذا الوضع ما يُعرَف «بحدود المتغيرات» (SCope of Variables) في لغات البرمجة حيث إنَّ أيَّ متغير في دالة

(FUNCTION) أو إجراء (PROCEDURE) يُقصَد به المتغير المعرَّف ضِمن الدالة أو الإجراء، وليس المتغيِّر الذي يتبع للجزء من البرنامج الذي قام بتنشيط الدالة أو الإجراء.

8–2–1 العوامل العلاقية IN, ANY, ALL:

يُلاحَظ في الاستفسارات الداخلية السابقة أنها تقومُ بـإعادة قيمة واحدة فقط. وحيث إنَّ الحالة العامة لنتائج الاستفسارات في لغة الاستفسار البنائية هي إعادة جداول تحتوي على مجموعات من القِيم، وليس قيمة واحدة فقط؛ فإن لغة الاستفسار البنائية توفر ثلاثة عوامل علاقية أخرى للتعامل مع مجموعات من القيم بالإضافة لعوامل المقارنة أعلاه التي تسبق الاستفسارات الفرعية. وهذه العوامل هي: IN, ANY, ALL.

يُستخدَم العامل العلاقي «ALL»؛ لمقارنة الحقل (أو التعبير الحسابي) قيد التحقق في عبارة الشرط لتعليمة الاستفسار الخارجية مع كافة القِيم الناتجة من الاستفسار الداخلي. ويجب أن يسبق هذا العامل أحد عوامل المقارنة. فعلى سبيل المثال: لمعرفة أعضاء هيئة التدريس الذين تزيد رواتبهم عن متوسطات المرتبات في كافة أقسام الجامعة، وترتيب النتيجة تصاعدياً حسب أسماء عائلات أعضاء هيئة التدريس، يمكِن استخدامُ التعليمة التالية:

```
SELECT *
FROM FACULTY_T
WHERE Salary > ALL (SELECT AVG(Salary)
FROM FACULTY_T
GROUP BY Department_ID)
ORDER BY LName;
```

ويمكِن فهم عمل التعليمة السابقة، كما يلي:

1- حساب متوسط رواتب أعضاء هيئة التدريس في كلِّ قسم (حسب تعليمة الاستفسار الداخلية).

2- لكلِّ عضو هيئة تدريس في جدول أعضاء هيئة التدريس (في الاستفسار الخارجي)؛ تتمُّ مقارنة راتبه بكافة متوسطات رواتب أعضاء هيئة التدريس لكافة أقسام الجامعة.

3- إذا كان راتب عضو هيئة التدريس أعلى من كافة متوسطات الرواتب؛ يتمُّ اختيار سجل عضو هيئة التدريس ضِمن نتيجة التعليمة.

4- يتمُّ اختيارُ الحقول المحدَّدة في تعليمة الاختيار الخارجية؛ لتمثِّل النتيجة النهائية للتعليمة (وهي كافة الحقول في هذه الحالة لاستخدام علامة '*').

5- تُرتَّب السجلات التي تمَّ اختيارها أبجدياً حسب أسماء عائلات أعضاء هيئة التدريس.

وتكون النتيجة النهائية للتعليمة، كما يلي:

FACULTY_	FNAME	LNAME	PHONE_NO	SALARY	DOB	DEPART
800	Ali	Albader	456-7812	45300	22-JUN-66	EE
420	Saleh	Alghamdi	454-2233	44600	13-FEB-69	CHEM
330	Ghanim	Alghanim	456-2234	44500	12-AUG-69	CS
320	Mohammed	Alhamad	454-5412	44000	13-MAY-65	CS
770	Sultan	Aljasir	456-3212	43300	13-MAY-70	PHYS
660	Saud	Alkhalifa	454-9856	44900	13-AUG-72	STAT
640	Fahad	Alzaid	456-3322	44300	12-MAY-71	STAT
810	Saad	Alzhrani	454-5578	44200	17-OCT-67	EE

تجدرُ الإشارة إلى أن نتيجة عامل المقارنة (ALL) ستتحقق، بغض النظر عما يسبق هذا العامل من عوامل المقارنة مثل '=' أو '<>' إذا كانت نتيجة الاستفسار الداخلي مجموعة خالية من القيم (EMPTY SET). ويعني هذا أنَّ الحقل (أو التعبير الحسابي) قيد التحقق منه بعامل المقارنة (ALL) سيحقق الشرط، ويكون ضِمن النتيجة النهائية للتعليمة إذا كانت نتيجة الاستفسار الداخلي مجموعة خالية من القِيم.

على النقيض من عامل المقارنة (ALL)؛ فإن العامل العلاقي (ANY) يشترط أن يكون قيمة الحقل أو التعبير الحسابي قيد التحقق مقترناً بـقيمة واحدة على الأقل من مجموعة القِيم الناتجة من الاستفسار الداخلي. وكما هو الحال في عامل المقارنة (ALL)؛ يجب أن يسبق عامل المقارنة (ANY) أحد عوامل المقارنة التالية: (<>, <, >, =, >=, <=). فعلى سبيل المثال: لمعرفة

أعضاء هيئة التدريس الذين تزيد رواتبهم عن متوسطات المرتبات في أيٍّ من أقسام الجامعة، وترتيب النتيجة تصاعدياً حسب أسماء عائلات أعضاء هيئة التدريس، يمكِن استخدامُ التعليمة التالية:

```
SELECT *
FROM FACULTY_T
WHERE Salary > ANY (SELECT AVG(salary)
FROM FACULTY_T
GROUP BY Department_ID)
ORDER BY LName;
```

وتكون نتيجة التعليمة السابقة، كما يلي:

FACULTY_	FNAME	LNAME	PHONE_NO	SALARY	DOB	DEPART
800	Ali	Albader	456-7812	45300	22-JUN-66	EE
560	Salman	Albassam	454-7865	33800	13-SEP-68	ENGL
420	Saleh	Alghamdi	454-2233	44600	13-FEB-69	CHEM
330	Ghanim	Alghanim	456-2234	44500	12-AUG-69	CS
320	Mohammed	Alhamad	454-5412	44000	13-MAY-65	CS
540	Salem	Alhamad	456-3304	40000	11-SEP-72	ENGL
770	Sultan	Aljasir	456-3212	43300	13-MAY-70	PHYS
660	Saud	Alkhalifa	454-9856	44900	13-AUG-72	STAT
400	Ahmad	Alotaibi	454-4563	33900	17-MAY-71	CHEM
200	Khalid	Aloufi	454-2341	35000	22-MAY-63	MATH
850	Ahmad	Alsabti	456-0120	33900	15-APR-73	EE
710	Mahmood	Alsalem	456-3323	31900	19-FEB-73	PHYS
640	Fahad	Alzaid	456-3322	44300	12-MAY-71	STAT
810	Saad	Alzhrani	454-5578	44200	17-OCT-67	EE
500	Yahya	Khorshid	456-2221	36700	12-MAR-65	ENGL

إنَّ نتيجة عامل المقارنة (ANY)، على النقيض من عامل المقارنة (ALL)؛ لا تتحقق، بغض النظر عمَّا يسبق هذا العامل من عوامل المقارنة مثل ‘=’ أو ‘< >’، إذا كانت نتيجة الاستفسار الداخلي مجموعة خالية من القيم (EMPTY SET). ويعني هذا أن الحقل (أو التعبير الحسابي) قيد التحقق منه بعامل المقارنة (ANY) لن يحقق الشرط؛ وبذلك لن يكون ضِمن النتيجة النهائية للتعليمة إذا كانت نتيجة الاستفسار الداخلي مجموعة خالية من القِيم.

أمَّا عامل المقارنة (IN)؛ فيعمل على التحقق من أن قيمة الحقل أو التعبير الحسابي قيد التحقق؛ هو من ضِمن مجموعة القِيم الناتجة من الاستفسار الداخلي. وبذلك؛ فهو مكافئ لعامل المقارنة (ANY) مسبوقاً بعامل المقارنة (=)، كما يلي: '=ANY'. ويعني هذا أن عامل المقارنة (ANY) أشمل في الاستخدام من عامل المقارنة (IN)؛ غير أن عامل المقارنة (IN) لا يتطلب أن يُسبق بعوامل مقارنة أخرى كما هو الحال في حالة عامل المقارنة (ANY)، وعامل المقارنة (ALL). فعلى سبيل المثال: لمعرفة أعضاء هيئة التدريس الذين لا يعملون في قسم الهندسة الكهربائية ويتقاضون مرتبات تساوي أيٍّ من أعضاء هيئة التدريس الذين يعملون في قسم الهندسة الكهربائية، وترتيب النتيجة تصاعدياً حسب أسماء عائلات أعضاء هيئة التدريس، يمكِن استخدام التعليمة التالية:

```
SELECT *
FROM FACULTY_T
WHERE Salary IN (SELECT DISTINCT Salary
CULTY_T     FROM FA
   WHERE Department_ID = 'EE')
AND Department_ID <> 'EE'
ORDER BY LName;
```

وتكون نتيجة التعليمة السابقة كما يلي:

```
FACULTY_ FNAME        LNAME        PHONE_NO     SALARY DOB       DEPART
-------- ------------ ------------ -------- ---------- --------- ------
400      Ahmad        Alotaibi     454-4563      33900 17-MAY-71 CHEM
```

وتكون نتيجة التعليمة السابقة مُكافئة لنتيجة استخدام عامل المقارنة (ANY) الواردة في التعليمة التالية:

```
SELECT *
FROM FACULTY_T
```

```
WHERE Salary = ANY (SELECT DISTINCT
Salary
CULTY_T           FROM FA
        WHERE Department_ID = 'EE')
AND Department_ID <> 'EE'
ORDER BY LName;
```

كما يُمكِن أن يُسبق عامل المقارنة (IN) بالنفي؛ ليصبح (NOT IN). ويعني هذا أن الحقل أو التعبير الحسابي قيد التحقق سيكون من ضِمن النتيجة النهائية للاستفسار الخارجي إذا لم يكن من ضِمن مجموعة القيم الناتجة من الاستفسار الداخلي قيمةٌ تكافئ قيمة الحقل أو التعبير الحسابي. ويكافئ عامل المقارنة (NOT IN) عامل المقارنة (ALL) مسبوقاً بعامل المقارنة (< >)، كما يلي: '< > ALL'. فعلى سبيل المثال: لمعرفة مواد الحاسب الآلي غير المنفذة في فصل الربيع (SPRING) من عام 2000؛ يمكن استخدام التعليمة التالية:

```
SELECT *
FROM COURSE_T
WHERE Course_ID NOT IN (SELECT
Course_ID
CTION_T                 FROM SE
     WHERE Semester = 'SPRING' AND
Year = '2000')
AND Course_ID LIKE 'CS%';
```

تقوم تعليمة الاستفسار الداخلية، أعلاه، باستخدام جدول المجموعات الدراسية لتحديد كافة المواد المنفذة في فصل الربيع من عام 2000. أمَّا تعليمة الاستفسار الخارجية فتقوم بتحديد مواد

الحاسب الآلي غير المنفذة في فصل الربيع من عام 2000 باستخدام عامل المقارنة (NOT IN) مع مجموعة القيم الناتجة من الاستفسار الداخلي. وتكون نتيجة التعليمة السابقة، كما يلي:

```
COURSE_ TITLE                               UNITS DEPART
------- ----------------------------------- ---------- ------
CS101   JAVA PROGRAMMING                          3 CS
CS104   COMPUTER ARCHITECTURE                     3 CS
CS105   INTRODUCTION TO DATABASE SYSTEMS          3 CS
```

8-2-2 الاستفسارات المتداخلة المتعددة المستويات:

تُمكِّن لغة الاستفسار البنائية تكوين استفسارات متداخلة بمستويات تزيد عن مستويين. فعلى سبيل المثال: لمعرفة أعضاء هيئة التدريس المؤهلين لتدريس أكثر من مادة بحيث إن كل مادة من المواد المؤهلين لتدريسها قد تمَّ تنفيذها مرةً واحدة على الأقل، يمكن استخدام الاستفسار ذي المستويات الثلاثة التالي:

```
SELECT *
FROM FACULTY_T
WHERE FaCulty_ID IN (SELECT FaCulty_ID
CATION_T          FROM QUALIFI
Course_ID IN (SELECT Course_ID                            WHERE
CTION_T)                          FROM SE
Culty_ID                          GROUP BY Fa
COUNT(FaCulty_ID) > 1);                       HAVING
```

ويتمُّ تنفيذ التعليمة السابقة كما يلي:

1- يتمُّ تحديد أرقام المواد الدراسية المنفذة أولاً؛ من خلال تنفيذ تعليمة الاستفسار الداخلية المعرَّف مصدر استرجاعها على أنه جدول المجموعات الدراسية.

2- يتمُّ تنفيذ تعليمة الاستفسار الأعلى في المستوى، والتي تحَدِّد أرقام أعضاء هيئة التدريس المؤهلين لتدريس مواد تقع ضمن مجموعة المواد المنفذة. ويتمُّ تجميع السجلات الناتجة حسب أرقام أعضاء هيئة التدريس. بعد ذلك؛ يتمُّ ترشيح المجموعات حسب عدد السجلات في كلِّ مجموعة؛ بحيث إن كلَّ مجموعة تمثل عدد تكرارات رقم عضو هيئة التدريس. وعندما يتكرر رقم عضو هيئة التدريس (في المجموعة) يكون رقم عضو هيئة التدريس ضِمن نتيجة الاستفسار الفرعي.

3- يتمُّ تنفيذ الاستفسار الأعلى في المستوى (وهو الاستفسار الأخير)؛ بحيث يتمُّ إظهار كافة بيانات أعضاء هيئة التدريس الذين تقع أرقامهم ضِمن مجموعة السجلات الناتجة من الاستفسار الفرعي الأدنى في المستوى.

وتكون النتيجة النهائية للاستفسار السابق، كما يلي:

```
FACULTY_ FNAME        LNAME        PHONE_NO     SALARY DOB       DEPART
-------- ------------ ------------ -------- ---------- --------- ------
200      Khalid       Aloufi       454-2341      35000 22-MAY-63 MATH
220      Fahad        Alhamid      456-7733      25900 07-OCT-70 MATH
320      Mohammed     Alhamad      454-5412      44000 13-MAY-65 CS
```

3-2-8 الاستفسارات المتداخلة المرتبطة (Correlated Nested Queries):

تمَّ شرحُ الاستفسارات المتداخلة التي تقوم لغة الاستفسار البنائية بتنفيذها مرةً واحدة فقط؛ بمعنى أن كلَّ استفسار داخلي يتمُّ تنفيذه مرةً واحدةً وبمعزل عن الاستفسار الخارجي الذي يحتويه. ويعني هذا أيضاً أن الاستفسارات الفرعية السابقة تُنفَّذ على أنها استفسارات مستقلة قائمة بذاتها دون أيِّ ارتباط بسجلات جدول الاستفسار الخارجي الذي يحتويها. وبعد تنفيذ الاستفسار الداخلي تُستخدَم نتيجته للمقارنة مع كلِّ سجل من سجلات الجدول المُستخدَم في تعليمة الاستفسار الخارجية. غير أنه في الكثير من الأحيان تظهر حاجةٌ لربط الاستفسار الداخلي بسجلات جدول (أو جداول) الاستفسار الخارجي. ويستدعي هذا عملية تنفيذ الاستفسار الداخلي لكلِّ سجل من سجلات جدول الاستفسار الخارجي. وتتمُّ عملية الارتباط هذه عندما يتمُّ استخدام أحد حقول جداول الاستفسار الخارجي ضِمن شرط الاسترجاع في الاستفسار الداخلي. في هذه الحالة يُسمَّى الاستفساران (الداخلي والخارجي) مترابطين (Correlated). وعلى النقيضِ من الاستفسارات غير المترابطة؛ يتمُّ استخدام بيانات السجلات الموجودة في جدول الاستفسار الخارجي في تنفيذ الاستفسار الداخلي.

وتتمُّ عملية التنفيذ لكلِّ سجل على حِدَة. فعلى سبيل المثال: لمعرفة أعضاء هيئة التدريس المؤهلين لتدريس مواد معيَّنة ولم يقوموا حتى الآن بتدريسها على الرغم من تنفيذها، يمكن استخدام التعليمة التالية:

```
SELECT *
FROM FACULTY_T
WHERE FaCulty_ID IN (SELECT FaCulty_ID
FROM QUALIFICATION_T Q
WHERE Course_ID NOT IN (SELECT Course_ID
FROM SECTION_T S
WHERE Q.FaCulty_ID = S.
FaCulty_ID)
Course_ID IN (SELECT Course_ID      AND
CTION_T));                          FROM SE
```

ويُمكِن فهم طريقة تنفيذ التعليمة السابقة، كما يلي:

1- تُنفَّذ تعليمة الاستفسار الداخلية غير المرتبطة؛ لمعرفة أرقام كافة المواد الدراسية التي تمَّ تنفيذها في الجامعة. وهذه التعليمة كما يلي:

```
SELECT Course_ID
FROM SECTION_T;
```

وتكون نتيجة التعليمة السابقة، كما يلي:

```
COURSE_
-------
CHEM101
CHEM101
CS101
CS101
CS102
CS103
CS104
CS105
EE101
EE102
ENGL101
ENGL102
MATH101
MATH102
MATH103
MATH104
PHYS101
PHYS102
STAT101
STAT102
```

وعلى الرَّغم من تكرار بعض أرقام المواد الدراسية في النتيجة؛ فإن هذا ليس ذا أهميةٍ على النتيجة النهائية لهذا المثال.

2- تُنفَّذ التعليمة المرتبطة الداخلية؛ لمعرفة أرقام المواد الدراسية التي قام كلُّ عضو هيئة تدريس بـتدريسها. فعلى سبيل المثال: لنفترض عضو هيئة التدريس رقم '660'. في هذه الحالة لن ينتج أيُّ سجل بعد تنفيذ التعليمة المرتبطة؛ لأن عضو هيئة التدريس هذا لم يقم بـتدريس أية مادة دراسية. ويكون شكلُ التعليمة المرتـبطة في هذه الحالة، كما يلي:

```
SELECT Course_ID
FROM SECTION_T S
WHERE '660' = S. FaCulty_ID;
```

3- تُنفَّذ تعليمة الاستفسار الخارجية، التي تدمجُ بين التعليمتين السابقتين؛ لمعرفة أرقام أعضاء هيئة التدريس المؤهلين لتدريس مواد دراسية؛ ولكنهم لم يدرسوا هذه المواد المؤهلين لتدريسها على الرغم من تنفيذها، كما يلي:

```
SELECT FaCulty_ID
FROM QUALIFICATION_T Q
```

```
WHERE Course_ID NOT IN (SELECT Course_ID
CTION_T S                          FROM SE
Culty_ID =                                   WHERE Q.Fa
S. FaCulty_ID)
AND Course_ID IN (SELECT Course_ID
CTION_T);                          FROM SE
```

وتكون نتيجة التعليمة، كما يلي:

```
FACULTY_
--------
770
660
```

ويُلاحَظ ظهور رقم عضو هيئة التدريس '660' وذلك لكون عضو هيئة التدريس هذا مؤهل لتدريس مادة الإحصاء (STAT101) حسب محتويات جدول المؤهلات التدريسية لأعضاء هيئة التدريس؛ ولكنه لم يقم بتدريسها حسب محتويات جدول المجموعات الدراسية؛ إذ قام عضو هيئة تدريس آخر (وهو ذو الرقم '600') بتدريس هذه المادة. وكذلك هو الحال بالنسبة لعضو هيئة التدريس رقم '770' المؤهل لتدريس مادة الفيزياء (PHYS101)؛ ولكنه لم يقم بتدريسها على الرغم من تنفيذها حسب جدول المجموعات الدراسية؛ حيث قام بتنفيذها عضو هيئة التدريس ذو الرقم '710'.

4- تُنفَّذ تعليمة الاستفسار الخارجية الأخيرة؛ لمعرفة بيانات أعضاء هيئة التدريس ذوي الأرقام الواردة ضِمن النتيجة النهائية (السابقة) للاستفسار الداخلي. وتكون نتيجة التعليمة، كما يلي:

```
FACULTY_ FNAME        LNAME        PHONE_NO     SALARY DOB       DEPART
-------- ------------ ------------ -------- ---------- --------- ------
660      Saud         Alkhalifa    454-9856      44900 13-AUG-72 STAT
770      Sultan       Aljasir      456-3212      43300 13-MAY-70 PHYS
```

8-2-3-1 العامل العلاقي EXISTS:

يُستخدَم عامل المقارنة EXISTS؛ لاختبار نتيجة الاستفسار الفرعي من حيث خلوه من أية نتيجة (EMPTY SET). فعندما يكون الاستفسار الفرعي خالياً من النتائج تصبح نتيجة العامل EXISTS خطأ (False). وعندما تحتوي نتيجة الاستفسار الفرعي على نتائج؛ تصبح نتيجة العامل EXISTS صح (True). وفي الغالبية العظمى من الأحيان يُستخدَم العامل العلاقي EXIXTS مع الاستفسارات المتداخلة المرتبطة. فعلى سبيل المثال: لمعرفة أعضاء هيئة التدريس الذين قاموا بتدريس مواد دراسية، يمكِن استخدام الاستفسار المتداخل المرتبط التالي:

```
SELECT *
FROM FACULTY_T F
WHERE EXISTS (SELECT *
              FROM SECTION_T S
              WHERE F.Faculty_ID = S.Faculty_ID);
```

يقوم الاستفسارُ السابق بتحديد سجل أحد أعضاء هيئة التدريس من جدول أعضاء هيئة التدريس الوارد في الاستفسار الخارجي، ومن ثمَّ يقوم باسترجاع كافة السِّجلات من جدول المجموعات الدراسية التي يُوجَد فيها رقم عضو هيئة التدريس الذي تمَّ تحديده من ضِمنها. بعد ذلك يتمُّ التحقق من خلوِّ نتيجة الاستفسار الداخلي من السجلات. وعندما تكون نتيجة الاستفسار الداخلي غير خالية من السجلات؛ فإن هذا يعني أن عضو هيئة التدريس قد قام بتدريس مواد دراسية. وخلاف ذلك؛ يعني أن عضو هيئة التدريس لم يقم بتدريس أية مادة دراسية. وتكون نتيجة التعليمة السابقة، كما يلي:

FACULTY_	FNAME	LNAME	PHONE_NO	SALARY	DOB	DEPART
200	Khalid	Aloufi	454-2341	35000	22-MAY-63	MATH
220	Fahad	Alhamid	456-7733	25900	07-OCT-70	MATH
310	Saleh	Aleesa	454-8932	30000	13-SEP-66	CS
320	Mohammed	Alhamad	454-5412	44000	13-MAY-65	CS
330	Ghanim	Alghanim	456-2234	44500	12-AUG-69	CS
340	Ibraheem	Alsaleh	454-1234	25000	20-JAN-70	CS
400	Ahmad	Alotaibi	454-4563	33900	17-MAY-71	CHEM
500	Yahya	Khorshid	456-2221	36700	12-MAR-65	ENGL
540	Salem	Alhamad	456-3304	40000	11-SEP-72	ENGL
600	Turki	Alturki	456-7891	27800	23-JUL-75	STAT
640	Fahad	Alzaid	456-3322	44300	12-MAY-71	STAT
710	Mahmood	Alsalem	456-3323	31900	19-FEB-73	PHYS
730	Mishal	Almazid	454-2343	29800	17-SEP-75	PHYS
800	Ali	Albader	456-7812	45300	22-JUN-66	EE
810	Saad	Alzhrani	454-5578	44200	17-OCT-67	EE

أمَّا إذا أردنا معرفة أعضاء هيئة التدريس الذين لم يقوموا بتدريس أية مادة دراسية؛ فيمكن استخدام الاستفسار المتداخل التالي الذي يُستخدَم فيه عامل المقارنة (NOT EXIXTS)، كما يلي:

```
SELECT *
FROM FACULTY_T F
WHERE NOT EXISTS (SELECT *
CTION_T S                    FROM SE
Culty_ID =                   WHERE F.Fa
S.FaCulty_ID);
```

وتكون نتيجة الاستفسار السابق، كما يلي:

FACULTY_	FNAME	LNAME	PHONE_NO	SALARY	DOB	DEPART
420	Saleh	Alghandi	454-2233	44600	13-FEB-69	CHEM
560	Salman	Albassam	454-7865	33800	13-SEP-68	ENGL
660	Saud	Alkhalifa	454-9856	44900	13-AUG-72	STAT
770	Sultan	Aljasir	456-3212	43300	13-MAY-70	PHYS
850	Ahmad	Alsabti	456-0120	33900	15-APR-73	EE

وكمثالٍ آخر؛ لنفترض أننا نرغب في معرفة المواد الدراسية التي لم تُنفَّذ من قِبـل الجامعة الأهلية حتى الآن. للإجابة عن هذا الاستفسار يمكِن استخدام التعليمة التالية:

```
SELECT *
FROM COURSE_T C
WHERE NOT EXISTS (SELECT *
CTION_T S                    FROM SE
C.Course_ID =        WHERE
S.Course_ID);
```

وتكون نتيجة التعليمة السابقة، كما يلي:

```
COURSE_ TITLE                                        UNITS DEPART
------- ----------------------------------- ---------- ------
CHEM102 CHEMISTRY (II)                                   3 CHEM
EE103   ELECTRONICS (II)                                 3 EE
EE104   COMMUNICATION NETWORKS                           4 EE
ENGL103 TECHNICAL WRITING                                3 ENGL
MATH106 ALGEBRA                                          4 MATH
MATH107 COMPUTER MATHEMATICS                             3 MATH
```

3–8 تعليمات الإضافة، والحذف، والتحديث (Insert, Delete and Update Statements):

توفر لغة الاستفسار البنائية ثلاث تعليمات لتغيير محتوى قاعدة البيانات؛ وهي: تعليمة الإضافة (Insert)، وتعليمة الحذف (Delete)، وتعليمة التحديث (Update). وفيما يلي شرح كلٍّ واحدة من هذه التعليمات الثلاث.

1–3–8 تعليمة الإضافة (Insert):

إن أبسط صور تعليمة الإضافة؛ هي: إضافة سجل لجدول ما. في هذه الحالة يتوجب ذِكر اسم الجدول، وقِيم الحقول للسجل قيد الإضافة. كما يتوجَّب إدخال قِيم الحقول وفق نفس الترتيب الذي تمَّ إدراجها فيه في أثناء إنشاء الجدول. فعلى سبيل المثال: يمكن إضافة بيانات عضو هيئة تدريس جديد يعمل في قسم الرياضيات، كما يلي:

```
INSERT INTO FACULTY_T
VALUES ('205', 'Saleh', 'Altimimi', '454-2233', 35000, '22-05-1963',
'MATH');
```

وتجدر ملاحظة القيود في أثناء عملية إدخال البيانات، مثل المفتاح الرئيسي الذي يجب ألا يتكرر مع سجل موجود أصلاً في الجدول، وقيد القيم الغائبة (Not Null)؛ إذ إن أيَّ حقل مرتبط بهذا القيد يجب أن تدخل قيمة له، وإلا فشلت عملية الإضافة، والقيد الفريد (Unique)؛ إذ إنَّ أيَّ حقلٍ مرتبطٍ بهذا القيد يجب أن تكون له قيمةٌ غير متكررة مع سجل موجود أصلاً في الجدول، على الرغم من أن قيمة الحقل من الممكن أن تكون غائبةً (NULL) لأكثر من سِجل واحد كما أسلفنا عند شرح قيود السجلات، وإلا فشلت عملية الإضافة أيضاً.

أمَّا الشكل الثاني لتعليمة الإضافة؛ فهو عند الرَّغبة في إدخال قِيم بعض الحقول، وليس جميعها. ويُعَدُّ هذا الشكلُ من التعليمة مفيداً جداً عندما يكون عددُ حقول الجدول كبيراً جداً ونرغب في إدخال بعضٍ منها فقط. وفي هذه الحالة يتوجب ذِكر أسماء الحقول التي سيتمُّ إدخال قِيم لها قبل إدراج القِيم التي سيتمُّ إدخالها. ونظراً لذكر أسماء الحقول ضِمن عملية الإضافة؛ فإنه يمكن إدخالها بأيِّ ترتيب شِئنا طالما أن القيم المُدخَلة متوافقة مع ترتيب الحقول. فعلى سبيل المثال: يمكِن إدخال بيانات عضو هيئة تدريس جديد يعملُ في قسم الرياضيات، كما يلي:

```
INSERT INTO FACULTY_T (FName, FaCulty_ID, LName,
Department_ID, DOB)
VALUES (‘Mohamed’, ‘207’, ‘Alsalem’, ‘MATH’, ‘22-05-1963’);
```

وعند استخدام الشكل السابق للتعليمة؛ فإن أيَّ حقلٍ لم تدخل له قيمة ضِمن تعليمة الإضافة ستكون قيمته غائبةً أو ستأخذ القيمة الافتراضية في حالِ ارتبط الحقل الذي لم تدخل قيمة له بقيمة افتراضية في أثناء إنشاء الجدول. أمَّا إذا أردنا أن تكون قيمة حقل ما غائبةً بشكلٍ صريح؛ فتستخدم كلمة "NULL" لهذا الغرض. فمثلاً: يمكِن أن نضع قيمة حقل الراتب (SALARY) غائبة بشكل صريح، كما يلي:

```
INSERT INTO FACULTY_T (FName, FaCulty_ID, LName,
Department_ID, DOB, Salary)
VALUES (‘Mohamed’, ‘207’, ‘Alsalem’, ‘MATH’, ‘22-05-1963’,
NULL);
```

ويُوجَد شكلٌ ثالث لعملية الإضافة يُستخدَم لإضافة مجموعة من السجلات، تكون ناتجة من عملية اختيار، إلى جدول. فعلى سبيل المثال: لنفترض أننا نرغب في إنشاء جدول جديد بمُسمَّى (TEMP_T) يحتوي على ثلاثة حقول: حقل يحتوي على رمز القسم الدراسي، وحقل يحتوي على عدد أعضاء هيئة التدريس في القسم، وحقل يحتوي على مجموع رواتب أعضاء هيئة التدريس في القسم. في هذه الحالة يمكن إنشاؤه، كما يلي:

```
CREATE TABLE TEMP_T
PRIMARY KEY,      CHAR(6)    (DEPARTMENT_ID
NUMBER,FACULTY_NO
NUMBER);TOTAL_SALARY
```

وبعد إنشاء الجدول، يمكِن إدخال البيانات إليه، كما يلي:

```
INSERT INTO TEMP_T
SELECT Department_ID, COUNT(*) FaCulty_NO, SUM(Salary)
Total_Salary
FROM FACULTY_T
GROUP BY Department_ID;
```

وتكون نتيجة العملية السابقة، كما يلي:

```
DEPART FACULTY_NO TOTAL_SALARY
------ ---------- ------------
CHEM            2        78500
CS              4       143500
EE              3       123400
ENGL            3       110500
MATH            2        60900
PHYS            3       105000
STAT            3       117000
```

كما يمكِن إدراج أسماء الحقول ضِمن تعليمة الإدخال، كما يلي:

```
INSERT INTO TEMP_T (Department_ID, FaCulty_NO,
Total_Salary)
SELECT Department_ID, COUNT(*), SUM(Salary)
FROM FACULTY_T
GROUP BY Department_ID;
```

وتجدر الإشارة إلى أنه ليس من الضروري أن يكون الجدول المُراد إدخال قِيم له فارغاً من السجلات؛ حتى يمكن استخدام الشكل السابق للتعليمة، وإنما قد يكون الجدول محتوياً على سجلات

قبل إدخال قِيم جديدة فيه باستخدام الشكل السابق للتعليمة.

2–3–8 تعليمة الحذف (Delete):

تُستخدَم تعليمة الحذف (DELETE)؛ لحذف السِّجلات من الجداول. وتحتوي تعليمة الحذف على عبارة الشرط (WHERE) الشبيه بشرط الاسترجاع في عبارة الاختيار (SELECT). ويتمُّ تنفيذ عملية الحذف على سجلٍ واحدٍ أو مجموعة من السجلات في جدول واحد فقط. فعلى سبيل المثال: يُمكِن حذف سجل عضو هيئة التدريس ذي الرقم ‘207’ الذي تمَّ إدخاله في جدول أعضاء هيئة التدريس أعلاه، كما يلي:

```
DELETE FROM FACULTY_T WHERE
FaCulty_ID=‘207’;
```

كما يمكِن حذف مجموعةٍ من السجلات عوضاً عن سجلٍ مُحدَّدٍ واحدٍ حسب الشرط الذي يتمُّ تحديده في عبارة الشرط. فمثلاً؛ يمكِن حذف الأقسام الدراسية التي يقلُّ عدد أعضاء هيئة التدريس فيها عن ثلاثة من الجدول المؤقت (TEMP_T)، كما يلي:

```
DELETE FROM TEMP_T WHERE FaCulty_NO < 3;
```

أمَّا إذا أردنا حذف كافة السِّجلات من جدولٍ ما؛ فإنه يمكِن استخدام تعليمة الحذف دون استخدام عبارة الشرط. فمثلاً؛ يمكِن حذف بقية السجلات الموجودة في الجدول المؤقت (TEMP_T)، كما يلي:

```
DELETE FROM TEMP_T;
```

وعلى الرَّغم من أننا قد قمنا بحذف كافة السِّجلات من الجدول المؤقت؛ فإن تعريف هيكل الجدول يستمرُّ موجوداً ضِمن مكوِّنات قاعدة البيانات. ويعني هذا أن تعليمة الحذف تقوم بحذف السجلات، سواء بشكلٍ فردي أو كمجموعات؛ ولكنها لا تقوم بإزالة هياكل الجدول. ولحذف هيكل

الجدول تُستخدَم تعليمة الإزالة (DROP)؛ كونها هي الوحيدة القادرة على إزالة هياكل البيانات، وبقية مكوِّنات قاعدة البيانات (مثل: القيود، والمنظورات). ولإزالة هيكل الجدول المؤقت، على سبيل المثال، تُستخدَم التعليمة التالية:

```
DROP TABLE TEMP_T;
```

وعند استخدام تعليمة الحذف؛ تجدر ملاحظة قيود السلامة المرجعية؛ إذ إنَّ حذف سجلٍ ما قد يؤدي إلى حذف سجلات في جداول أخرى مرتبطة بالسجل موضع الحذف بـقيود المفتاح الخارجي. وحسب تعريف القيود الخارجية، كما أسلفنا سابقاً، إمَّا أن تقبل عملية الحذف ويتخذ الفعل المناسب إزاء المفاتيح الخارجية هذه (سواء وضع قيمها بحيث تكون غائبةً أو لقيمة افتراضية ما)، أو ترفض عملية الحذف (عند ارتباط أحد المفاتيح الخارجية برد الفعل "RESTRICT")، أو يتمُّ حذف السجل وكافة السجلات التي تشير له ضِمن مفاتيحها الخارجية (عندما تكون مرتبطةً بردِّ الفعل "CASCADE").

8-3-3 تعليمة التحديث (Update):

تُستخدَم تعليمة التحديث (UPDATE)؛ لتحديث قِيم حقول سجل واحد أو أكثر في جدولٍ ما. وكما هو الحال في تعليمة الحذف السابقة؛ تُستخدَم العبارة الشرطية (WHERE) لتحديد السجلات التي ستُجرَى عليها عملية التحديث. فمثلاً: لتحديث راتب عضو هيئة التدريس ذي الرقم '200' ليصبح 40000 (عوضاً عن 35000)؛ يمكِن استخدام تعليمة التحديث، كما يلي:

```
UPDATE FACULTY_T
SET Salary = 40000
WHERE FaCulty_ID = '200';
```

ويمكِن استخدامُ التعليمة لتحديث مجموعةٍ من السِّجلات، عوضاً عن سجل واحد؛ بحيث ينطبق عليها شروط عبارة "WHERE". فمثلاً: يمكِن استخدام التعليمة التالية لزيادة مرتبات أعضاء هيئة التدريس الذين يعملون في قسم الرياضيات بنسبة 10%:

```
UPDATE FACULTY_T
SET Salary = Salary * 1.1
WHERE Department_ID =
‘MATH’;
```

كما يُمكِن تحديث كافة السجلات في الجدول؛ وذلك عند عدم استخدامنا للعبارة الشرطية (WHERE) ضِمن تعليمة التحديث. فمثلاً: يمكِن زيادة رواتب كافة أعضاء هيئة التدريس في الجامعة الأهلية بنسبة 10% وفق تعليمة التحديث التالية:

```
UPDATE FACULTY_T
SET SALARY = Salary * 1.1;
```

وكما هو الحالُ عند استخدام تعليمة الحذف، تجدر ملاحظة قيود السلامة المرجعية في تعليمة التحديث؛ إذ إن تحديث قيمة المفتاح الرئيسي لسجلٍ ما قد يترتب عليه تحديث المفاتيح الخارجية المعرَّفة في جداول أخرى تشيرُ للسجل موضع التحديث. في هذه الحالة تتحَدَّد عملية التحديث على السجل من عدمها وفق الضوابط الموضوعة على المفاتيح الخارجية والمصاحبة لعبارة «عند التحديث» (ON UPDATE) في الجداول التي تشير إلى السجل ضِمن مفاتيحها الخارجية، كما أسلفنا سابقاً. فعلى سبيل المثال: لو حاولنا تغيير المفتاح الرئيسي لعضو هيئة التدريس رقم ‘200’ ليصبح ‘205’؛ فإن هذه العملية ستفشل بمعنى أن نظام إدارة قاعدة البيانات لن يقوم بتنفيذها. والسبب وراء ذلك يعود إلى كون بعض السجلات المدوَّنة في جدول المجموعات الدراسية ترتبط بمفاتيح خارجية تشير إلى رقم عضو هيئة التدريس هذا على أساس أنه الشخص الذي يقوم بتدريس هذه المجموعات الدراسية، وأننا لم نربط المفتاح الخارجي في جدول المجموعات بعبارة «عند التحديث» (On Update). وبالتالي؛ فإن الوضع الافتراضي عند التحديث هو التقييد (RestriCt)، أيْ: عدم التحديث.

```
UPDATE FACULTY_T
SET FaCulty_ID = ‘205’
```

```
WHERE FaCulty_ID = '200';
```

ويُمكِن استخدامُ تعليمة التحديث لتحديث أكثر من حقل (وفي سجل أو أكثر من السجلات). فمثلاً: يمكِن استخدام التعليمة التالية؛ لزيادة مرتبات أعضاء هيئة التدريس الذين يعملون في قسم الحاسب الآلي "CS" بنسبة 10% ونقلهم للعمل في قسم الرياضيات "MATH".

```
UPDATE FACULTY_T
SET Salary = Salary * 1.1,
Department_ID = 'MATH'
WHERE Department_ID = 'CS';
```

وتكون نتيجة التعليمة السابقة عند تنفيذها في بيئة أوراكل، كما يلي:

```
SQL> UPDATE FACULTY_T
  2  SET SALARY = SALARY * 1.1, DEPARTMENT_ID = 'MATH'
  3  WHERE DEPARTMENT_ID = 'CS';

4 rows updated.
```

وباستعراض جدول أعضاء هيئة التدريس؛ نجد أن رواتب وأقسام أعضاء هيئة التدريس الأربعة الذين يعملون في قسم الحاسب الآلي، وهم ذوو الأرقام الوظيفية (310، 320، 330، 340)، قد تمَّ تحديثها؛ بحيث تمَّت زيادة رواتبهم بنسبة 10%، وأصبح القسم الذين يتبعونه هو قسم الرياضيات "MATH" عوضاً عن قسم الحاسب الآلي "CS"، كما يلي:

FACULTY_	FNAME	LNAME	PHONE_NO	SALARY	DOB	DEPART
200	Khalid	Aloufi	454-2341	35000	22-MAY-63	MATH
220	Fahad	Alhamid	456-7733	25900	07-OCT-70	MATH
310	Saleh	Aleesa	454-8932	33000	13-SEP-66	MATH
320	Mohammed	Alhamad	454-5412	48400	13-MAY-65	MATH
330	Ghanim	Alghanim	456-2234	48950	12-AUG-69	MATH
340	Ibraheem	Alsaleh	454-1234	27500	20-JAN-70	MATH
400	Ahmad	Alotaibi	454-4563	33900	17-MAY-71	CHEM
420	Saleh	Alghamdi	454-2233	44600	13-FEB-69	CHEM
500	Yahya	Khorshid	456-2221	36700	12-MAR-65	ENGL
540	Salem	Alhamad	456-3304	40000	11-SEP-72	ENGL
560	Salman	Albassam	454-7865	33800	13-SEP-68	ENGL
600	Turki	Alturki	456-7891	27800	23-JUL-75	STAT
640	Fahad	Alzaid	456-3322	44300	12-MAY-71	STAT
660	Saud	Alkhalifa	454-9856	44900	13-AUG-72	STAT
710	Mahmood	Alsalem	456-3323	31900	19-FEB-73	PHYS
730	Mishal	Almazid	454-2343	29800	17-SEP-75	PHYS
770	Sultan	Aljasir	456-3212	43300	13-MAY-70	PHYS
800	Ali	Albader	456-7812	45300	22-JUN-66	EE
810	Saad	Alzhrani	454-5578	44200	17-OCT-67	EE
850	Ahmad	Alsabti	456-0120	33900	15-APR-73	EE

8–4 دوال الوقت والتاريخ، ودوال الأرقام، ودوال السلاسل الحرفية، ودوال التحويل:

توفِّر لغة الاستفسار البنائية مجموعةً من الدوال التي تمكِّن من معالجة البيانات المخزَّنة في قاعدة البيانات. ولاستكمال شرح تعليمات لغة الاستفسار البنائية في بيئة أوراكل نشرح في هذا الجزء بعضاً من هذه الدوال. وتُسمَّى هذه الدوال في بعض الأحيان بدوال الصفوف (Row FunCtions)؛ وذلك للتفريق بينها وبين دوال التجميع (التي تُسمَّى أحياناً دوال الأعمدة)، وسبق أن قمنا بشرحها في الجزء (7-2-1-8).

8–4–1 دوال الوقت والتاريخ:

تُخزَّن بيانات الوقت والتاريخ في بيئة أوراكل كبيانات رقمية لتمثيل ما يلي:

CENTURY	القرن
YEAR	السنة
MONTH	الشهر
DAY	اليوم
HOURS	الساعات
MINUTES	الدقائق
SECONDS	الثواني

والصيغة الضِّمنية لإدخال وعرض التاريخ هي (DD-MON-YY)؛ بحيث إن (DD) تمثل تاريخ اليوم، و(MON) تمثل الثلاثة أحرف الأولى من الشهر، و(YY) تمثل السنة، كما في '20-JUL-99'. وتُستخدَم الدالة (SYSDATE) لاسترجاع تاريخ اليوم من نظام التشغيل؛ وذلك باستخدام جدول افتراضي (Dummy) مُخصَّص لهذا الغرض في بيئة أوراكل هو (SYS.DUAL). فعلى سبيل المثال: يمكِن استعراض تاريخ اليوم، كما يلي:

```
SELECT SYSDATE
FROM SYS.DUAL;
```

وتكون نتيجة التعليمة السابقة، كما يلي:

```
SYSDATE
--------
03/08/18
```

ويُمكِن إجراء عمليات مختلفة على الوقت والتاريخ من ضِمنها جمعُ عدد على تاريخ، وطرح عددٍ من تاريخ، وطرح تاريخ من تاريخ، كما يلي:

1- لحساب تاريخ الغد: SYSDATE + 1.

2- لحساب تاريخ الأمس: SYSDATE – 1.

3- لحساب الوقت بعد ست ساعات: 24/SYSDATE + 6؛ بحيث إن عدد ساعات اليوم هو 24 ساعة.

4- لحساب الوقت بعد عشر دقائق: SYSDATE + 10/1440؛ بحيث إن عدد الدقائق في اليوم هو 1440 دقيقة.

5- لحساب الوقت بعد عشر ثوان: SYSDATE + 10/86400؛ بحيث إن عدد الثواني في اليوم هو 86400 ثانية.

كما يُمكِن استخدامُ الدوال التالية للتعامل مع الوقت والتاريخ:

1- لإضافة أو طرح عدد(n) من الشهور من تاريخ (date) طبقاً لإشارة (n (±)) تُستخدَم الدالة التالية:

ADD_MONTHS (date, n)

2- لإيجاد فرق الشهور بين تاريخين؛ بحيث يكون الناتج سالباً إذا كان التاريخ (date1) أصغر من (date2)، كما قد يحتوي الناتج على جزءٍ عشري يمثل فرق الأيام بين التاريخين، تُستخدَم الدالة التالية:

MONTHS_BETWEEN (d1, d2)

3- لتقريب التاريخ والوقت طبقاً لشكل (Format) معين، ويكون التقريب إلى أقرب سنة، أو شهر أو أيِّ جزءٍ من أجزاء التاريخ والوقت، ومع إهمال (Format) يكون التقريب إلى منتصف ليل أقرب يوم، تستخدم الدالة التالية:

ROUND (date[,format])

4- لاستقطاع جزءٍ من التاريخ والوقت طبقاً لشكل ((Format معين، ومع إهمال ((Format يكون الوقت هو الصفر (أيْ: منتصف الليل) 12.00AM، تُستخدَم الدالة التالية:

```
TRUNC (date[,format])
```

5- لإيجاد تاريخ آخر يوم من الشهر الذي يقع فيه التاريخ ((date، تُستخدَم الدالة التالية:

```
LAST_DAY (date)
```

فعلى سبيل المثال: لمعرفة تاريخ آخر يوم من شهر مارس لعام 1999م، تُستخدَم التعليمة التالية:

```
SELECT LAST_DAY ('05-03-99')
FROM DUAL;
```

وتكون نتيجة التعليمة السابقة، كما يلي:

```
LAST_DAY
--------
31/03/99
```

8–4–2 دوال الأرقام:

توفِّر لغة الاستفسار البنائية في بـيئة أوراكل مجموعةً من الدوال التي يُمكِن استخدامها مع الأرقام. وهذه الدوال، كما يلي:

1- لتقريب حقل رقمي (A1) إلى حقل رقمي يحتوي على (A2) خانة على يمين الفاصلة العشرية، وبحيث يكون الناتج عددًا صحيحاً إذا كانت (A2=0).

```
ROUND (A1,[A2])
```

ومن استخدامات الدالة السابقة؛ خفضُ عددِ الأرقام العشرية الناتجة من استخدام دوال الأعمدة (أو التجميع) في جدول النتائج.

2- لتمثيل حقل رقمي (A1) يحتوي على (A2) خانة على يمين الفاصلة العشرية، تستخدم الدالة التالية:

TRUNC (A1,[A2])

3- لإرجاع باقي قسمة العدد (A1) على العدد (A2)؛ تُستخدَم الدالة التالية:

MOD (A1,A2)

4- لإرجاع القيمة المطلقة للعدد (A1)، مع ملاحظة أن القيمة المطلقة دائماً موجبة، تُستخدَم الدالة التالية:

ABS (A1)

5- لمعرفة إشارة العدد (A1)؛ بحيث تكون النتيجة هي(1) إذا كان A1 موجباً و (-1) إن كان سالباً، و(0) إذا كان العدد يساوي صفراً، تُستخدَم الدالة التالية:

SIGN (A1)

فعلى سبيل المثال: لمعرفة متوسط مرتبات أعضاء هيئة التدريس الذين يعملون في قسم الفيزياء، ومتوسط مرتبات أعضاء هيئة التدريس الذين يعملون في قسم الكيمياء مقربين إلى رقمين عشريين، وإلى أقرب عددين صحيحين؛ نستخدم التعليمة التالية:

```
SELECT Department_ID, AVG(Salary), ROUND(AVG(Salary),2),
TRUNC(AVG(Salary))
FROM FACULTY_T
WHERE Department_ID = ‘PHYS’ OR Department_ID = ‘CHEM’
GROUP BY Department_ID;
```

وتكون نتيجة التعليمة السابقة، كما يلي:

```
DEPART AVG(SALARY) ROUND(AVG(SALARY),2) TRUNC(AVG(SALARY))
------ ----------- -------------------- ------------------
CHEM         39250                39250              39250
PHYS         35000                35000              35000
```

8-4-3 دوال السلاسل الحرفية:

1- لاختيار جزءٍ من سلسلة حرفية (A1) ابتداءً من الموقع (A2)، وبحيث تمثل (A3) عدد الحروف المطلوبة في السلسلة الحرفية الناتجة، مع ملاحظة أنه عند حذف (A3) تحتوي النتيجة على كلِّ حروف (A1) التي على يمين (A2)، تُستخدَم الدالة التالي:

```
SUBSTR (A1, A2 [,A3])
```

فعلى سبيل المثال: تكون نتيجة التعليمة التالية الحروف الأربعة الأولى من السلسلة الحرفية ابتداءً من أول حرف فيها:

```
SELECT SUBSTR (‘1994-01-01’, 1,
4)
FROM DUAL;
```

وتكون نتيجة التعليمة السابقة، كما يلي:

2- لإيجاد طول الحقل (A1) بما في ذلك الفراغات والأصفار السابقة (Leading Zeroes)، تُستخدَم الدالة التالية:

```
LENGTH (A1)
```

3- لربط سلسلتين حرفيتين قد تكونا قيماً في عمودين، أو قيمة عمود وثابت حرفي كعمود واحد، تُستخدَم الدالة التالية:

```
A1 || A2
```

4- لتحويل جميع حروف سلسلة حرفية (إنجليزية) إلى حروفٍ كبيرة، تُستخدَم الدالة التالية:

```
UPPER (A1)
```

5- لتحويل جميع حروف سلسلة حرفية (إنجليزية) إلى حروف صغيرة؛ تُستخدَم الدالة التالية:

```
LOWER (A1)
```

6- لتحويل أول حرف من كلِّ كلمة (إنجليزية) في سلسلة حرفية إلى حرفٍ كبير؛ بحيث تكون الفواصل بين الكلمات هي المسافة (SPACE) أو أحد الرموز التالية (. : ؛ # ! $ أو غيرها)، تُستخدَم الدالة التالية:

```
INITCAP (A1)
```

4-4-8 دوال التحويل:

1- تتعاملُ البرامج مع الشكل الخارجي للتاريخ كسلسلةٍ حرفية، ويتمُّ تحويله مباشرةً إلى الصيغة الضمنية. وتُستخدَم الدالة التالية لتحويل تاريخ (date) إلى سلسلة حرفية طبقاً لشكل (Format) معين:

TO_CHAR (date, format)

وفيما يلي بعض الأشكال القياسية:

DATE	TIME	FORMAT
yyyy-mm-dd	hh.mm.ss	ISO
mm/dd/yyyy	hh:mm PM or hh:mm AM	USA
dd.mm.yyyy	hh.mm.ss	EUR

وبإهمال الشكل (Format)؛ يتمُّ تحويل التاريخ طبقاً للصيغة الضمنية. وفيما يلي الأشكال (Format) المختلفة لصيغة التاريخ:

DD	**رقم اليوم من الشهر (1 إلى 31).**
DAY	لعرض اسم اليوم كاملاً (Sunday to Friday) في 9 خانات.
DY	لعرض اسم اليوم مختصراً .(Sun to Fri)
MM	رقم الشهر من السنة (1 إلى 12).
MONTH	لعرض اسم الشهر كاملاً (January to DeCember) في 9 خانات.
MON	لعرض اسم الشهر مختصراً .(Jan to DeC)
YY	أول رقمين من السنة، 98 مثلاً.
YYYY	رقم السنة كاملاً، 1998 مثلاً.

CC	رقم القرن الميلادي.
HH or HH12	الساعة من 1 إلى 12.
AM or PM	لتحديد الوقت ما إذا كان قبل أو بعد منتصف الليل.
HH24	الساعة من 1 إلى 24.
MI	الدقيقة من 1 إلى 60.
SS	الثانية من 1 إلى 60.
; : - / ,	علامات التوقيف.
“text”	نص داخل علامات تنصيص.
TH	رتبة الأرقام كما في .(1st, 2nd, 3rd, 4th, 5th, ...)
SP	الأرقام كتابة .(first, seCond, …)
FM	أسماء الأيام والشهور دون إضافة فراغات .(Blank Padding)

ومن الأمثلة التطبيقية على تحويل التاريخ إلى سلاسل حرفية، ما يلي:

طريقة التحويل	النتيجة
TO_CHAR(SYSDATE, ‘fmMonth, ddth,yyyy’)	May, 12th, 1998
TO_CHAR(SYSDATE, ‘Month, ddsp,yyyy’)	May , twelve, 1998
TO_CHAR(SYSDATE, ‘“On the” ddspth “of” fmMONTH “at” hh:mi:ssPM’)	On the Twelfth of MAY at 11:34:29AM

2- لتحويل الرقم (Number) إلى سلسلة حرفية، تستخدم التعليمة التالية:

TO_CHAR (Number [,format])

بحيث يمكِن أن يكون الشكل (Format) على إحدى الصيغ التالية:

99990	عدد التسعات والأصفار يحَدِّد عدد الخانات الممكن عرضها.
999,999.99	يمكِن استخدامُ الفاصلة والفاصلة العشرية؛ للتحكُّم في طريقة العرض.
$999	لعرض الرقم كعملة.
S999	لعرض الإشارة (- أو +) قبل الرقم.
999S	لعرض الإشارة (- أو +) بعد الرقم.
999MI	لعرض (-) بعد الرقم إذا كان الرقم سالباً. لا تظهر الإشارة الموجبة.
RN	لعرض الرقم بشكل الأرقام الرومانية.

3- لتحويل سلسلة حرفية إلى تاريخ طبقاً للشكل (Format)، وبإهمال الشكل (Format) يجب أن تكون السلسلة الحرفية مطابقةً للصيغة الضمنية (dd-mon-yyyy). يمكن استخدام جميع أشكال (Format) المُستخدَمة مع دالة (TO_CHAR) عدا (th, sp, fm ,"text")، تستخدم الدالة التالية:

TO_DATE (string [,format])

ومن الأمثلة التطبيقية على تحويل السلاسل الحرفية إلى تواريخ، ما يلي:

النتيجة	طريقة التحويل
12-May-98	TO_DATE('12-MAY-98')
12-May-98	TO_DATE('May, 12, 1998', 'Month, dd, yyyy')

5-8 لغة التحكُّم في البيانات (Data Control Language (DCL)):

توفِّر نظمُ إدارة قواعد البيانات العلاقية إمكانيةَ التحكُّم في الصلاحيات المخوَّلة للمستفيدين للتعامل مع البيانات المخزَّنة في قاعدة البـيانات. وحيث إنه يمكِن التعامل مع قاعدة الـبيانات الواحدة من قِبل أكثر من مُستخدِم وفي آنٍ واحد؛ فإن نظم إدارة قواعد البيانات العلاقية تمكِّن من التعامل المتزامن للبيانات من قِبل أكثر من مُستخدِم؛ وذلك عن طريق تزويد كلِّ مُستخدِم لنظام إدارة قاعدة البيانات برمزٍ خاص يُمكِّنه من التعامل مع قاعدة البـيانات. ويُستخدَم الرمز الخاص بكلِّ مُستخدِم في تعريف الصلاحيات التي تخوِّله للتعامل مع الجداول المختلفة المعرَّفة في قاعدة البـيانات. وتوفر لغة الاستفسار الـبنائية تعليمتين يُمكِن من خلالهما التحكُّم في الصلاحيات التي تُعطَى للمستفيدين للتعامل مع قاعدة البيانات أو سحب الصلاحيات منهم. وهاتان التعليمتان هما: تعليمة مَنْح الصلاحية (Grant)، وتعليمة سحب الصلاحية (Revoke).

8–5–1 مَنْح الصلاحيات:

عند إنشاء جدول جديد باستخدام تعليمة الإنشاء (Create)؛ يكون الجدول المنشأ مِلكاً للمُستخدِم (أو المستفيد) الذي قام بـإنشائه. ولمعرفة مُلاك الجداول المختلفة في قاعدة الـبيانات؛ تقوم نظم إدارة قواعد البيانات العلاقية (داخلياً) بوَضْع رمز المستخدِم الذي يملك الجدول قبـل اسم الجدول. فلو افترضنا أن رمز المستخدِم الذي قام بإنشاء جدول المواد الدراسية (Course_T) هو "Houmaily"؛ فإن نظام إدارة قاعدة البيانات سيقوم بتسمية الجدول داخلياً بنفس الاسم الذي أُستخدِم في تعليمة إنشاء الجدول مسبوقاً باسم المستخدِم الذي قام بإنشائه، كما يلي:

HOUMAILY.COURSE_T

وتعني الطريقة السابقة في تسمية الجداول داخل قاعدة البيانات؛ أنه بالإمكان إنشاء أكثر من جدول بنفس المُسمَّى؛ ولكن من قِبـل مستخدمين (أو مستفيدين) مختلفين. ويقوم نظام إدارة قواعد الـبيانات في حَلِّ أيِّ التباسٍ قد يظهر نتيجةً لتكرار مُسمَّيات الجداول من خلال إدراج رمز المستفيد الذي يملِك الجدول قبـل اسم الجدول، كما هو أعلاه. وعند الرجوع لجدول يتكرَّر اسمه ضِمن جداول قاعدة الـبيانات؛ فإنه يتعَيَّن على المستفيد إدراج رمز مالك الجدول قبل اسم الجدول؛ حتى يتمكَّن نظام إدارة قاعدة البيانات من التعرُّف على الجدول المقصود دون أيِّ التباس مع الجداول

الأخرى التي تحمل نفس المُسمَّى. وفي حالة عدم التقيُّد بذلك من قِبل المستفيد؛ فإن نظام إدارة قاعدة البيانات لن يتمكَّن من التعرُّف على الجدول المقصود. وبالتالي؛ فإنه لن يقوم بتنفيذ العملية الصادرة من قِبل المستفيد التي تحتوي على اسم جدول يتكرَّر مع مُسمَّيات جداول أخرى في قاعدة البيانات. أمَّا في حالة عدم وجود التباس في مُسمَّيات الجداول؛ فإنه يُمكِن استخدام اسم الجدول مباشرةً دون إدراج رمز المستخدِم الذي يملك الجدول.

وباستطاعة المستفيد الذي قام بإنشاء جدول ما التعامل مع هيكل الجدول والبيانات المخزَّنة فيه؛ إذ إنَّ بإمكانه حذف هيكل الجدول أو التعديل عليه كما أن بإمكانه استرجاع البيانات الموجودة في الجدول والتعديل عليها (من خلال عمليات الحذف، والإضافة، والتحديث). ونظراً؛ لأنه بالإمكان مَنْح صلاحيات مُحدَّدة على أيِّ جدول لمستفيدين آخرين خلاف الشخص المالك للجدول الذي يملك الصلاحيات الكاملة للتعامل مع هيكل الجدول ومحتوياته؛ فإن نظام إدارة قاعدة البيانات يقوم بتحديد الصلاحيات الممنوحة لكلِّ مستخدِم وتدوينها في ملفات خاصة بالنظام. وتُمكِّن نظم إدارة قواعد البيانات العلاقية من إعطاء الصلاحيات التالية للمستفيدين على جداول قاعدة البيانات:

- صلاحية الاسترجاع (SeleCt Privilege): تُمكِّن هذه الصلاحية من استرجاع البيانات المخزَّنة في الجدول باستخدام تعليمة الاختيار (أو الاسترجاع) (SeleCt Statement).

- صلاحية الإضافة (Insert Privilege): تُمكِّن هذه الصلاحية من إضافة سجلات جديدة للجدول باستخدام تعليمة الإضافة (Insert Statement).

- صلاحية الحذف (Delete Privilege): تُمكِّن هذه الصلاحية من حذف سجلات موجودة في الجدول باستخدام تعليمة الحذف (Delete Statement).

- صلاحية التحديث (Update Privilege): تُمكِّن هذه الصلاحية من تعديل القيم المخزَّنة في سجلات الجدول باستخدام تعليمة التحديث (Update Statement).

وعند إنشاء جدول جديد يكون للمستفيد الذي قام بإنشاء الجدول وحده كافة الصلاحيات المُدرَجة أعلاه؛ في حين لا يمتلك أيُّ مستفيدٍ آخر أية صلاحية للتعامل مع محتويات الجدول. وحتى يتمكَّن مستفيدٌ آخر من التعامل مع محتويات الجدول؛ فإنه لا بد وأن يقوم مالك الجدول بمَنْح المستفيد

بعض الصلاحيات المُدرَجة أعلاه. ولمَنْح المستفيد صلاحية التعامل مع الجدول؛ تُستخدَم تعليمة (Grant) التي تأخذ الشكل العام التالي:

```
GRANT Privileges ON Table_Name TO User;
```

ويُقصَد بـ (Privileges) الصلاحيات الممنوحة، و(Table_Name) اسم الجدول الذي ستُمنَح عليه الصلاحيات، و(User) رمز المستفيد الذي سيتمُّ منحه للصلاحيات. فعلى سبيل المثال: يُمكِن مَنْح صلاحية الاسترجاع للمستفيد (Student1) على جدول المواد الدراسية من قِبل مالك الجدول، وليكن المستفيد (Houmaily)، كما يلي:

```
GRANT SELECT ON COURSE_T TO
STUDENT1;
```

ويُمكِن أيضاً مَنْح أكثر من صلاحية، من خلال نفس تعليمة منح الصلاحية؛ للتعامل مع الجدول من قِبل المستفيد، كما يلي:

```
GRANT SELECT, UPDATE, DELETE ON COURSE_T TO
STUDENT1;
```

وباستطاعة المستفيد (Student1) الآن تنفيذ تعليمات الاسترجاع والتحديث والحذف على جدول المواد الدراسية. وتوفر لغة الاستفسار البنائية إمكانية مَنْح كافة الصلاحيات المُدرَجة أعلاه بشكلٍ مختصرٍ دون الحاجة إلى سَرْد الصلاحيات الواحدة تلو الأخرى ضِمن تعليمة مَنْح الصلاحية؛ وذلك من خلال استخدام عبارة (All Privileges)، كما يُوضِّح المثال التالي:

```
GRANT ALL PRIVILEGES ON COURSE_T TO STUDENT1;
```

ويُمكِن أيضاً الاستغناء عن الكلمة الاختيارية (Privileges) من التعليمة أعلاه؛ لتصبح أكثر اختصاراً، كما يلي:

```
GRANT ALL ON COURSE_T TO STUDENT1;
```

وبعد تنفيذ التعليمة أعلاه؛ يصبح للمستفيد (Student1) كافة الصلاحيات التي يملكها المستفيد الذي قام بإنشاء الجدول، والتي تُمكِّنه من التعامل مع محتويات الجدول من خلال أية تعليمة من تعليمات لغة معالجة البيانات. ويُمكِن أيضاً مَنْح الصلاحيات لأكثر من مستخدِم في نفس الوقت. فعلى سبيل المثال: يُمكِن مَنْح صلاحية استرجاع بيانات جدول المواد الدراسية لكلٍّ من (Student2)، و(Student3)، و(Student4)؛ من خلال استخدام نفس تعليمة مَنْح الصلاحية، كما يلي:

```
GRANT SELECT ON COURSE_T TO STUDENT2, STUDENT3,
STUDENT4;
```

ولإعطاء صلاحية معينة على جدولٍ ما لكافة المستفيدين من قاعدة البيانات عوضاً عن مستفيدين مُحدَّدين تُستخدَم كلمة (PubliC)، أيْ: لعموم المستفيدين. فعلى سبيل المثال: يُمكِن مَنْح صلاحية استرجاع بيانات جدول المواد الدراسية لعموم المستفيدين، كما يلي:

```
GRANT SELECT ON COURSE_T TO PUBLIC;
```

وتعني التعليمة السابقة أنه بإمكان أيِّ مستفيدٍ أن يقوم بتنفيذ تعليمة الاسترجاع على جدول المواد الدراسية. كما يُمكِن استخدامُ كلمة (ALL) مع كلمة (PUBLIC) لمنح كافة الصلاحيات على جدولٍ ما ولكافة المستفيدين من قاعدة البيانات. فعلى سبيل المثال: يمكِن مَنْح كافة الصلاحيات لكافة المستفيدين على جدول المواد الدراسية، كما يلي:

```
GRANT ALL ON COURSE_T TO PUBLIC;
```

وتوفِّر تعليمة مَنْح الصلاحية إمكانية إعطاء الصلاحيات على حقول معينة في الجدول؛ عوضاً عن كافة حقوله. وبهذه الطريقة يُمكِن حَجْب التعامل مع بعض البيانات الحساسة في الجدول عن المستفيدين؛ مما يُقدِّم حمايةً أكثر دقةً للبيانات المخزَّنة في الجدول. فعلى سبيل المثال: يمكِن إعطاء صلاحية تحديث القسم الدراسي الذي تتبعه المادة الدراسية لكافة المستفيدين، مع حجب إمكانية تحديث أيٍّ من الحقول الأخرى في جدول المواد الدراسية، كما يلي:

```
GRANT UPDATE(Department_ID) ON COURSE_T TO PUBLIC;
```

وتجدر الإشارة إلى أن مقياس لغة الاستفسار البنائية ينصُّ على أنه بالإمكان تحديد الحقول التي بالإمكان مَنْح الصلاحية عليها عندما تكون الصلاحية المُعطاة هي صلاحية التحديث (Update). أمَّا بالنسبة للصلاحيات الأخرى؛ فإن مَنْح الصلاحية سيكون على كافة حقول الجدول؛ غير أن بعض نظم إدارة قواعد البيانات العلاقية تمكِّن من مَنْح الصلاحيات الأخرى (غير صلاحية التحديث) على أعمدة مُحدَّدة عوضاً عن كافة حقول الجدول.

8-5-1-1 مَنْح الصلاحيات على المنظورات:

يُوجَد للمنظورات، شأنها شأن الجداول، صلاحيات يُمكِن منحها، وكذلك حجبها. وكما هو الحال بالنسبة للجداول يمكِن مَنْح الصلاحيات باستخدام تعليمة (Grant). فعلى سبيل المثال: يُمكِن مَنْح صلاحية الاسترجاع لكافة المستفيدين على منظور جدول المواد الدراسية، على افتراض وجود مثل هذا المنظور ضِمن هياكل قاعدة البيانات، كما يلي:

```
GRANT SELECT ON COURSE_V TO PUBLIC;
```

غير أنه من الضرورة بمكان الإشارة إلى أن مَنْح صلاحية التحديث، والإضافة، والحذف تُعَدُّ أكثر تعقيداً؛ وذلك لأن المنظور قد لا يكون قابلاً للتحديث عليه، أو الإضافة إليه، أو الحذف منه كما سبق توضيحه سابقاً عند حديثنا عن المنظورات. وتُعزَى هذه المعضلة إلى تعريف المنظور في أثناء إنشائه باستخدام تعليمة إنشاء المنظور (Create View). وبناءً على تعريف المنظور؛ فإن

بعض عمليات مَنْح الصلاحية قد لا يمكن تنفيذها؛ لكونها تتضارب مع العمليات التي يُمكِن تنفيذها على المنظور نفسه. فعلى سبيل المثال: لا يمكن مَنْح صلاحية التحديث على منظور إذا كان معرفاً بطريقة لا تقبل بإجراء عمليات التحديث عليه، مثل احتوائه على دوال تقوم بتجميع البيانات (Aggregate FunCtions)، أو إن كان يدمج بين محتويات أكثر من جدول.

8-5-1-2 إعطاء الحق في تخويل الصلاحية:

تسمح لغة الاستفسار البنائية لمالك الجدول بإعطاء حقٍّ ممارسة تخويل الصلاحية لمستفيد (أو مجموعة من المستفيدين)؛ بمعنى أن يصبح لهذا المستفيد (أو مجموعة المستفيدين) القدرة على مَنْح الصلاحيات التي خُوِّلت لهم من قِبل مالك الجدول لمستفيدين آخرين. ويمكِن ممارسة هذا الحق من خلال استخدام عبارة (With Grant Option). فعلى سبيل المثال: يمكِن للمستفيد "Houmaily" مَنْح كافة الصلاحيات التي يملكها على جدول المواد الدراسية، الذي قام بإنشائه، للمستفيد "Student1" مع إعطائه الحق في تخويل الصلاحيات التي أُعطِيت له لأيِّ مستفيدين آخرين، كما يلي:

```
GRANT ALL ON COURSE_T TO STUDENT1
;WITH GRANT OPTION
```

وتعني عبارة "With Grant Option" الواردة في نهاية التعليمة السابقة أنَّ مالك الجدول وهو "Houmaily" قد أعطى المستفيد "Student1" الحقَّ في تخويل الصلاحيات الممنوحة له لمستفيدين آخرين؛ إضافةً إلى حقه في ممارسة الصلاحيات المعطاة له للتعامل مع محتويات الجدول. وبناءً على حق تخويل الصلاحية الذي مُنِح للمستفيد "Student1" من قِبل مالك الجدول؛ فإنه بإمكان المستفيد "Student1" تخويل بعض أو كل الصلاحيات التي تمَّ إعطاؤها له على الجدول لمستفيدين آخرين. ويعني هذا أنه من ضِمن الصلاحيات التي أُعطِيت للمستفيد "Student1"؛ صلاحية استخدام تعليمة مَنْح الصلاحية (Grant) على جدول المواد الدراسية بما يتوافق مع صلاحيات الاسترجاع والتعديل التي أعطيت له. وتجدر الإشارة إلى أنه ليس من الضروري أن تُعطَى كافة الصلاحيات لمستفيد ما مع صلاحية الحق في تخويل الصلاحية حتى يتمكَّن من ممارسة حقه في تخويل الصلاحية، كما هو الحال في المثال السابق. فعلى سبيل المثال: يمكِن مَنْح

المستفيد "Student1" الحقَّ في تنفيذ عمليات الاسترجاع فقط على جدول المواد الدراسية مع حق تخويل الصلاحية. وفي هذه الحالة يمكِن للمستفيد "Student1" تنفيذ تعليمات الاسترجاع على جدول المواد الدراسية وكذلك تخويل هذه الصلاحية فقط لمستفيدين آخرين.

2–5–8 سحب الصلاحيات:

حيث إنه بالإمكان مَنْح الصلاحيات للمستفيدين؛ فإنه بالإمكان كذلك سَحْب الصلاحيات منهم. وتُستخدَم تعليمة سَحْب الصلاحيات (Revoke) لسحب الصلاحيات الممنوحة للمستفيدين. وتعمل تعليمة سَحْب الصلاحية على سَحْب صلاحيات مُحدَّدة من المستفيدين مثلها مثل تعليمة مَنْح الصلاحية التي تعطي صلاحيات مُحدَّدة للمستفيدين. فعلى سبيل المثال: يمكن سحب صلاحية التحديث من المستفيد "Student1" الممنوحة له على جدول المواد الدراسية، كما يلي:

```
REVOKE UPDATE ON COURSE_T FROM STUDENT1;
```

أمَّا إذا ما أريد سحب صلاحية الإضافة، وصلاحية الحذف من المستفيد "Student1" الممنوحتين له على جدول المواد الدراسية؛ فإنه يمكِن تنفيذ تعليمة سَحْب الصلاحية التالية:

```
REVOKE INSERT, DELETE ON COURSE_T FROM STUDENT1;
```

كما يُمكِن أن تُستخدَم عبارة (ALL) ضِمن تعليمة سَحْب الصلاحية كاختصار يُقصَد به كافة الصلاحيات الممنوحة. فعلى سبيل المثال: يمكن سحب كافة الصلاحيات الممنوحة للمستفيد "Student1" باستخدام عبارة (ALL)، كما يلي:

```
REVOKE ALL ON COURSE_T FROM STUDENT1;
```

وكذلك يُمكِن استخدام عبارة (PubliC) كاختصار يُقصَد به كافة المستفيدين. فعلى سبيل المثال: يُمكِن سحب صلاحية الإضافة وصلاحية الحذف وصلاحية التحديث من كافة المستفيدين، كما يلي:

```
REVOKE INSERT, DELETE, UPDATE ON COURSE_T FROM
PUBLIC;
```

أمَّا إذا أريد حجب إمكانية التعامل مع جدول المواد الدراسية عن كافة المستفيدين؛ فإنه يمكِن تنفيذ تعليمة سحب الصلاحية التالية:

```
REVOKE ALL ON COURSE_T FROM PUBLIC;
```

الفصل التاسع

موضوعات متقدمة في نظم قواعد البيانات

يتطرَّق هذا الفصل، باقتضاب، إلى أربعة موضوعات متطوِّرة ومهمة في نظم قواعد البيانات، وهي: المعاملات، وقواعد البيانات الشيئية، وقواعد البيانات «العلاقية - الشيئية»، وقواعد البيانات الموزَّعة. تُمثِّل المعاملات الوسيلة الرئيسية التي يتمُّ من خلالها التفاعل مع قواعد البيانات من قِبل المستفيدين؛ سواء بشكلٍ تفاعلي مباشر أم من خلال برامج التطبيقات التي يقوم مُطوِّرو التطبيقات ببنائها. أمَّا نموذج البيانات الشيئي؛ فقد تمَّ تطويره لسدِّ الاحتياجات التقنية التي يتطلبها تطويرُ نظم التطبيقات المتعلقة بمكننة أعمال المنظمات ذات الصبغة غير التقليدية؛ من حيث البيانات التي تتعامل معها مثل استخدامها في تطبيقات التصميم بمساعدة الحاسب الآلي (Computer-Aided Design)، والتصنيع بمساعدة الحاسب الآلي (Computer-Aided ManufaCturing)، والتجارب العلمية، ونظم المعلومات الجغرافية (GeographiCal Information Systems)، وتطبيقات الوسائط المتعددة (Multimedia AppliCations)؛ على سبيل المثال لا الحصر.

ونظراً لانتشار النموذج العلاقي وسهولته في تمثيل البيانات والتعامل معها، عكف الكثيرُ من الشركات المصنِّعة لنظم إدارة قواعد البيانات العلاقية على تبنِّي بعض مفاهيم النموذج الشيئي ضمن منتجاتها؛ حتى تتمكَّن من مواكبة احتياجات المنظمات التي تتصف بياناتها بالصبغة غير التقليدية؛ إضافةً إلى تلك التي تتصف بالتقليدية. وأصبحت مثل هذه المنتجات تُسمَّى قواعد البيانات «العلاقية - الشيئية». أمَّا بالنسبة للمنظمات التي تتوزَّع فيها مقرَّاتها في مناطق عديدة، وعلى رُقع متباعدة جغرافياً في الكثير من الأحيان؛ فقد دفعت هذه المنظمات الباحثين إلى تبنِّي مفهوم النظم

الموزَّعة، وأضحت تُسمَّى في مجال نظم قواعد البيانات «نظم قواعد البيانات الموزَّعة». وتوفر مثل هذه النظم العديدَ من الميزات مقارنةً بتلك النظم المركزية من ضِمنها «الموثوقية» (Reliability) و«التواجد» (Availability)، هذا بالإضافة إلى أدائها المتميِّز وسهولة التوسُّع في الأجهزة والتطبيقات في مثل هذه المنظمات. ونظراً لأهمية المفاهيم الأربعة السابقة؛ كان من الضروري التطرُّق إليها في هذا الكتاب ولو بشكلٍ مقتضب.

1–9 المعاملات (TransaCtions):

تُعَدُّ المعاملات الوسيلة الرئيسية التي يتمُّ من خلالها التفاعل مع قواعد البيانات من قِبل المستفيدين؛ سواءٌ بشكلٍ تفاعلي مباشر أم من خلال برامج التطبيقات التي يقوم مُطوِّرو التطبيقات ببنائها. وتُعَرف المعاملة بأنها برنامج، أو جزء من برنامج، حاسوبي يتمُّ من خلاله التفاعل مع قاعدة بيانات؛ بحيث يقوم بتحويل قاعدة البيانات من حالة صحيحة إلى حالة أخرى صحيحة تتوافق مع الضوابط المفروضة على قاعدة البيانات. وقد يكون البرنامج الحاسوبي مُكوَّناً بالكامل من تعليمات تتفاعل مع قاعدة البيانات، مثل تعليمات لغة الاستفسار البنائية؛ وذلك عندما يتمُّ التفاعل مع قاعدة البيانات بشكلٍ مباشر (أو تفاعلي) (InteraCtive Mode) دون تضمين هذه التعليمات في برنامج مكتوب بلغة برمجة عامة (General Purpose Programming Language)؛ أو قد يكون البرنامج مُكوَّناً من تعليمات تتفاعل مع قاعدة البيانات، ولكن هذه التعليمات مكتوبة ضِمن ثنايا إحدى لغات البرمجة العامة (مثل: سي، وجافا، وكوبول... إلخ). وفي كلتا الحالتين يتكون أو يحتوي البرنامج على تعليمات يمكِن فهمها ومعالجتها من قِبل نظام إدارة قاعدة البيانات.

وتقوم كلُّ معاملة إذا تمَّ تنفيذها بشكلٍ منفرد على قاعدة البيانات دون تداخل مع أيَّة معاملات أخرى تحت التنفيذ على قاعدة البيانات نفسها، ودون أيَّة أعطال للنظام في أثناء تنفيذ المعاملة بنقل قاعدة البيانات من حالة سليمة إلى حالة سليمة أخرى لا تحتوي على بيانات تخترق أيًّا من القيود المفروضة على قاعدة البيانات. فعلى سبيل المثال: قد يحتوي البرنامج على بعضٍ من تعليمات لغة الاستفسار البنائية إذا كان البرنامج يتفاعل مع قاعدة بيانات علاقية. ويُعَدُّ كلُّ تنفيذ لمجمل مجموعة التعليمات الموجودة في البرنامج معاملةً واحدةً. وتنحصرُ التعليمات التي تتكوَّن منها أيَّة معاملة بين عملية «بداية» (Begin) تشير إلى بداية تنفيذ معاملة جديدة على قاعدة البيانات، وعملية «نهاية» (End) تشيرُ إلى انتهاء المعاملة. ويتمُّ إدراج عددٍ من عمليات التعديل (الإضافة،

والحذف، والتحديث) والاسترجاع على قاعدة البيانات بين تعليمتَيْ البداية والنهاية. وتُستخدَم عمليتا البداية والنهاية من قِبل نظام إدارة قاعدة البيانات للتعرُّف على بداية كلِّ معاملة ونهايتها، في حين تُستخدَم التعليمات الأخرى للتفاعل مع محتويات قاعدة البيانات. كما أن غالبية نظم إدارة قواعد البيانات تتعرَّف على بداية كلِّ معاملة بشكلٍ ضِمني؛ وذلك عند تنفيذ أول تعليمة تتفاعل مع قاعدة البيانات من قِبل أحد المستخدمين أو أحد نُظُم التطبيقات، في حين يتمُّ التعرُّف على نهاية المعاملة ضِمنياً أيضاً من خلال تنفيذ تعليمة التثبيت (Commit) أو تعليمة الانسحاب (RollbaCk)، التي تُسمَّى أيضاً تعليمة الإخفاق (Abort)، من قِبل المعاملة. وعندما تكون نهاية المعاملة تعليمة تثبيت؛ فإن هذا يعني أن المُستخدَم (أو التطبيق) يرغب في تثبيت جميع التعديلات التي أُجريَت من قِبل المعاملة التي قام بتنفيذها على قاعدة البيانات. أمَّا عندما تكون نهاية المعاملة تعليمة انسحاب (أو إخفاق)؛ فإن هذا يعني أن المستخدِم (أو التطبيق) يرغب في عدم تثبيت أيٍّ من التعديلات التي قامت المعاملة بتنفيذها على قاعدة البيانات؛ مما يعني إرجاع جميع قِيم البيانات التي تفاعلت معها المعاملة إلى ما كانت عليه قبل تنفيذ المعاملة كما لو أنه لم يتم تنفيذ المعاملة على محتويات قاعدة البيانات على الإطلاق. وتتصف المعاملات في قواعد البيانات بأربع خصائص، هي (Bernstein et al, 1987; Gray and Reuter, 1992):

1- النووية (AtomiCity): تعني هذه الخاصية أن كلَّ معاملة تُنفَّذ باعتبارها وحدة منطقية واحدة غير قابلة للتجزئة؛ بحيث إن كافة العمليات التي تحتويها المعاملة إما أن يتمَّ تنفيذها بالكامل على قاعدة البيانات، وإما ألَّا يتمَّ تنفيذُ أيٍّ منها على الإطلاق.

2- الصحة (أو التوافق) (ConsistenCy): تعني هذه الخاصية أن كلَّ معاملة عبارة عن جزءٍ من برنامج (حاسوبي) قد تمَّت كتابته بشكلٍ صحيحٍ يتوافق مع الضوابط التي وُضِعَت على قاعدة البيانات بالإضافة إلى الضوابط المرعية في المنظمة.

3- العُزلة (Isolation): تعني هذه الخاصية أن كلَّ معاملة يتمُّ تنفيذها بشكلٍ منعزل على قاعدة البيانات دون أيِّ تداخل مع المعاملات الأخرى التي قد تكون قيد التنفيذ بالتزامن معها على قاعدة البيانات نفسها.

4- الدوام (Durability): تعني هذه الخاصية أنَّ أيَّ تعديلات تُجرَى على قاعدة البيانات من قِبل المعاملات التي تنتهي، وتثبت نتائجها على قاعدة البيانات سيستمرُّ وجود نتائجها على قاعدة

البيانات حتى لو تعطل النظام مستقبلاً.

وتُعْرف الخصائصُ الأربع السابقة للمعاملات بمُسمَّى خصائص «أَسِد» (ACID Properties) اختصاراً لها؛ بحيث يمثل هذا المُسمَّى الحروف الأولى من مُسمَّيات الخصائص أعلاه. وعند بناء نظم التطبيقات تتمُّ كتابة المعاملات بداخل إحدى لغات البرمجة العامة، مثل: سي (C)، أو جافا (JAVA)، أو كوبول (COBOL)، أو داخل إحدى لغات البرمجة المخصَّصة لتطوير نظم التطبيقات التي تقوم الشركات المصنِّعة لنظم إدارة قواعد البيانات بتوفيرها لتطوير التطبيقات على نظم قواعد البيانات التي تقوم بتصنيعها، مثل: «أوراكل دفلوبر» (OraCle Developer) من شركة أوراكل. وتُسمَّى اللغة التي تحتوي على تعليمات تتعامل مع قاعدة البيانات باللغة المضيفة (Host Language). فعلى سبيل المثال: لو افترضنا وجود قاعدة بيانات تتعلق بنظام حجز للرحلات الجوية مكونة من ثلاثة جداول، كما يلي:

1- جدول الرحلات الجوية SRC, DEST, STSOLD, ,FNO, DATE)FLIGHT CAP) الذي يتكوَّن من: حقل رقم الرحلة (FNO)، وتاريخها (DATE)، ومحطة الإقلاع (SRC)، ومحطة الوصول (DEST)، وعدد المقاعد المباعة (STSOLD)، وسعة الطائرة (CAP).

2- جدول العملاء (أو الركاب) (ADDR, BAL ,CNAME)CUST الذي يتكون من: اسم العميل (CNAME)، وعنوانه (ADDR)، ورصيده (BAL).

3- جدول حجوزات العملاء (SPECIAL ,FNO, DATE, CNAME)FC الذي يتكوَّن من: حقل رقم الرحلة (FNO)، وتاريخها (DATE)، واسم العميل (CNAME)، واحتياجات العميل الخاصة (SPECIAL).

ومن ثم؛ فإنَّ البرنامج التالي، المكتوب بلغة سي (C)، يتضمَّن معاملةً تتفاعل مع قاعدة البيانات المعرَّفة أعلاه؛ بهدف إجراء حجوزات للعملاء على الرحلات الجوية.

يتكوَّن البرنامج من جزءٍ يتعلق بتعريف المتغيِّرات المشتركة التي تستخدمها كلٌّ من لغة البرمجة المضيفة ولغة الاستفسار البنائية. ويتمُّ وَضْع هذه المتغيرات بين بداية ونهاية ما يُعرَف بجزء توصيف المتغيرات المشتركة (DECLARE SECTION). كما يجب أن يتبع أية تعليمة

تتعاطى مع قاعدة البيانات بالكلمتين المحجوزتين (EXEC SQL)؛ وذلك للتفريق بين التعليمات الخاصة بلغة البرمجة والتعليمات التي تتعاطى مع قاعدة البيانات. وبهذه الطريقة يتمكَّن مترجمُ لغة البرمجة من تجاهل تعليمات لغة الاستفسار البنائية في أثناء ترجمة البرنامج (Program Compilation) وإرسالها بشكلٍ مباشرٍ لقاعدة البيانات في أثناء تنفيذ البرنامج. وبعد الجزء المتعلِّق بتوصيف المتغيِّرات المشتركة؛ تُوجَد تعليمة خاصة بلغة سي تهدف إلى قراءة المُدخلات التي من المفترض أن يقوم المستفيدُ بإدخالها للبرنامج، وهي: رقم الرحلة (flight_no)، وتاريخها (day)، واسم العميل (Customer_no). وبعد الحصول على بيانات العميل المزمَع إجراء حجز له من قِبل المستفيد (الذي يقوم بإجراء الحجز) يُنفذ البرنامج تعليمة تقوم بتحديث جدول الرحلات الجوية؛ بحيث يزيد عدد المقاعد المباعة بمقعد واحد؛ وذلك للرحلة الجوية المطلوب إجراء الحجز عليها. ويُلاحَظ هنا استخدام النقطتين المزدوجتين عند استخدام المتغيرات المشتركة ضِمن تعليمات لغة الاستفسار البنائية؛ وذلك حتى يتمكَّن معالج لغة الاستفسار البنائية من التفريق بين مُسمَّيات المتغيرات المشتركة وبقية الكلمات والعبارات الواردة في تعليمات لغة الاستفسار البنائية. بعد ذلك يقوم البرنامج بإضافة بيانات العميل والرحلة الجوية لجدول حجوزات العملاء. وينهي البرنامج المعاملة من خلال تنفيذ تعليمة التثبيت (Commit) وتعليمة الإطلاق (Release) التي تقوم بفصل المعاملة عن قاعدة البيانات، وإطلاق الموارد التي تمَّ حجزها في قاعدة البيانات. وتجدر الإشارة إلى أن البرنامج السابق غير مكتمل؛ إذ يجب الاتصال بقاعدة البيانات التي سيقوم البرنامج بالتعامل معها وتزويدها برقم المستخدِم وكلمة السر قبل التعامل الفعلي معها؛ ولهذا السبب قد تمَّ وَضْع علامات تنقيط في بداية البرنامج للدلالة على ذلك.

```
...
main {
...
    EXEC SQL BEGIN DECLARE SECTION;
    char flight_no [6], customer_name [20];
    char day;
    EXEC SQL END DECLARE SECTION;
    scanf (flight_no, day, customer_name);
    EXEC SQL UPDATE FLIGHT
       SET STSOLD = STSOLD + 1
       WHERE FNO = :flight_no AND DATE = :day;
    EXEC SQL INSERT
       INTO FC(FNO, DATE, CNAME, SPECIAL);
       VALUES(:flight_no, day, :customer_name, null);
    EXEC SQL COMMIT RELEASE;
    printf ("Reservation completed");
    return(0);}
```


ويُلاحَظ في المثال السابق أن تعليمة بدْء المعاملة جاءت ضِمنية؛ وذلك عند تنفيذ أول تعليمة من تعليمات لغة الاستفسار البـنائية. أمَّا تعليمة إنهاء المعاملة؛ فقد جاءت ضمن تعليمة التثبيت (Commit). ويمكِن أيضاً استخدامُ تعليمة الانسحاب في المعاملات عوضاً عن تعليمة التثبيت كما يوضِّح المثال التالي الذي يتعامل مع قاعدة البيانات السابقة نفسها.

```
...
main{
...
    EXEC SQL BEGIN DECLARE SECTION;
        char flight_no[6], customer_name[20];
        char day; int temp1, temp2;
    EXEC SQL END DECLARE SECTION;
    scanf(flight_no, day, customer_name);
    EXEC SQL SELECT STSOLD, CAP INTO :temp1, :temp2
        FROM FLIGHT
        WHERE FNO = :flight_no AND DATE = :day;
    if temp1 = temp2 then {
        printf("no free seats");
        EXEC SQL ROLLBACK RELEASE;
        return(-1);}
    else {
        EXEC SQL UPDATE FLIGHT
            SET STSOLD = STSOLD + 1
            WHERE FNO = :flight_no AND DATE = :day;
        EXEC SQL INSERT INTO
            FC(FNO, DATE, CNAME, SPECIAL)
            VALUES(:flight_no, :day, :customer_name, null);
        EXEC SQL COMMIT RELEASE;
        printf("Reservation completed");
        return(0);}
}
```

يقوم البرنامجُ السابق بنفس عمل برنامج الحجز الذي أسلفنا شرحه أعلاه؛ غير أنه يقوم بإجراء عملية قراءة مُسبقة لكلٍّ من عدد المقاعد المحجوزة، وعدد المقاعد التي تمثل السعة الكلية للرحلة الجوية التي من المفترض إجراء الحجز عليها؛ وذلك للتأكد من توفر مقاعد شاغرة لم يتم حجزها قبل إجراء أيّ حجز جديد. وللتأكد من ذلك تتمُّ قراءة عدد المقاعد المحجوزة وتخزينها في المتغير (temp1)، وتتمُّ قراءة سعة الطائرة التي ستقوم بالرحلة الجوية وتخزينها في المتغير (temp2). وبعد ذلك تتمُّ مقارنة المتغيرين من حيث تساويهما. وعند تساوي قيمة المتغيرين؛ فإن هذا يعني عدم توفُّر مقاعد شاغرة؛ لكون عدد المقاعد المباعة على الطائرة مساوياً لسعتها الكلية من الركاب؛ مما يستدعي إنهاء المعاملة من خلال عملية التراجع (أو الانسحاب) (ROLLBAK). أمَّا إذا لم يكن المتغيِّران متساويين؛ فإنه يمكِن إجراء عملية الحجز وتثبيت النتيجة على قاعدة البيانات كما في المثال السابق. ويمكِن تعديل البرنامج السابق؛ بحيث يقوم بإجراء حجوزات لأكثر

من مقعد؛ وذلك من خلال تعريف متغير مشترك جديد من نوع الأعداد الصحيحة، وليكن عدد المقاعد المطلوبة (no_of_seats)، وتعديل البرنامج بحيث يتواكب مع هذا التعديل.

1-1-9 التأكيد على خصائص المعاملات في نظم إدارة قواعد البيانات:

عند تنفيذ المعاملات على قواعد البيانات؛ فإنه لا بد أن تقوم نظم إدارة قواعد البيانات (Database Management Systems) بالمحافظة على خصائص المعاملات. ويتمُّ ذلك من خلال بناء نظامين فرعيين ضِمن أيّ نظام لإدارة قواعد البيانات، وهما: نظام التحكُّم في التزامن (ConCurrenCy Control ProtoCol)، ونظام الاستعادة (أو التشافي) (ReCovery ProtoCol). وعند تنفيذ المعاملات على قاعدة بيانات يقوم نظام إدارة قاعدة البيانات بتفكيك كلِّ تعليمة (تتفاعل مع قاعدة البيانات) من تعليمات أيَّة معاملة إلى تعليمات بسيطة تتكوَّن من عمليات قراءة (Read) وعمليات كتابة (Write)؛ بحيث إن كلَّ عملية بسيطة تتفاعل مع عنصرٍ واحد من عناصر قاعدة البيانات. وقد يكون العنصرُ حقلاً من حقول أحد سجلات جدول من جداول قاعدة البيانات، أو قد يكون سجلاً من سجلاتها، أو قد يكون كتلة (BloCk) من كتلِ جدول ما تمثل مجموعة من سجلاته، أو قد يكون حتى جدولاً كاملاً. ومع ذلك؛ فإن ماهية العنصر غير ذات أهمية من الناحية المنطقية عندما نتحدث عن نظام التحكم في التزامن ونظام الاستعادة.

9–1–1–1 نظام التحكُّم في التزامن (ConCurrenCy Control ProtoCol):

يتمُّ بناءُ نظام التحكُّم في التزامن؛ لضمان خاصية العُزلة (Isolation). ويُمْكِن ضمان هذه الخاصية من قِبل نظم إدارة قواعد البيانات؛ من خلال بناء نظام فرعي يقوم بجدولة العمليات البسيطة للمعاملات المختلفة؛ بحيث تبدو المعاملات التي تتبعها هذه العمليات وكأنها تُنفَّذ بالتسلسل الواحدة تلو الأخرى؛ على الرغم من تنفيذها بشكلٍ متزامن، أيْ: في الوقت نفسه، على قاعدة البيانات. وبهذه الطريقة تُنفَّذ المعاملات بشكلٍ متزامن على قاعدة البيانات، ولكن تأثيرها على قاعدة البيانات يظهر كأن المعاملات تُنفَّذ بشكل متسلسل الواحدة تلو الأخرى. وعلى افتراض أن قاعدة البيانات (التي تمَّ البدءُ منها) صحيحة، تحتوي على بيانات تتوافق مع القيود المفروضة عليها، وأن كلَّ معاملة قد تمَّت كتابتها بشكلٍ صحيح يتوافق مع الضوابط (أو القيود) المفروضة على قاعدة البيانات؛ فإن تنفيذ مجموعة من المعاملات على قاعدة البيانات بشكلٍ متزامن سينتج عنه حالة جديدة صحيحة أيضاً لقاعدة البيانات طالما أن المعاملات قد تمَّ تنفيذها بشكلٍ يتماثل مع تنفيذها بشكلٍ

متسلسلٍ ما، الواحدة تلو الأخرى دون أيِّ تداخل (أو تزامن) بـينها. وتُمكِّن نظرية التسلسل (Serializability Theory) من التحقق من أن أيَّ طريقة لضبـط التزامن تحقق خاصية العزلة من عدم تحقيقها لذلك. ومن أشهر الطرق المتـبعة في بناء نظم التحكُّم في التزامن التي تضمن تماثل تنفيذ المعاملات بشكلٍ متزامنٍ مع تنفيذها بشكلٍ متسلسل طريقة الأقفال (أو الحجز المسبق) (LoCking ProtoCols).

وباستخدام طريقة الأقفال؛ يتمُّ ربط كل عنصر من عناصر قاعدة البيانات بقفل؛ بحيث يمكن أن يُوضَع القفل في وَضْع مُخصَّص لقراءة العنصر، ويكون القفل في هذه الحالة في وَضْع حجز مشاركة أو قراءة (Shared or Read LoCk)، أو في وَضْع مُخصَّص للكتابة على العنصر، ويكون القفل في هذه الحالة في وَضْع حجز استثناء أو كتابة (ExClusive or Write LoCk). ونظراً؛ لأن أيَّ عمليتَيْ قراءة تابعتين لمعاملتين مختلفتين لا تؤثران على محتويات قاعدة البيانات؛ فإنه بالإمكان تنفيذهما بأيِّ ترتيب كان. ولهذا السـبب؛ فإن قفل المشاركة يسمحُ بأن تقوم أكثر من معاملة بقراءة العنصر نفسه في الوقت نفسه دون تضارب بـين العمليتين. وفي هذه الحالة؛ يُقال إن عمليات قراءة العناصر تتوافق (Compatible) مع بـعضها. وبناءً على ذلك؛ فإنه يمكِن تنفيذ عمليات القراءة على عناصر قاعدة البيانات بأيِّ ترتيب كان؛ بمعنى أنه يمكِن تنفيذها بشكلٍ تبادلي (Commutable) على عناصر قاعدة البـيانات دون التأثير على القِيم المُسترجَعة من العناصر. أمَّا عمليات الكتابة؛ فإنها تؤثر في محتويات قاعدة البيانات؛ ولذلك فإن ترتيب عمليات الكتابة على قِيم العناصر من قِبل المعاملات المختلفة مهمٌّ؛ إذ إن القيمة النـهائية للعنصر أو القيمة المُسترجَعة منه تتوقف على القيمة التي قامت بكتابتها آخر معاملة على العنصر. لذا؛ فإن ترتـيـب أيِّ عمليتي كتابة تابعتين لمعاملتين مختلفتين مُنفَّذتين على عنصرٍ ما يتضاربان؛ وذلك لكون ترتيبـهما يؤثر في القيمة النهائية للعنصر. وكذلك هو الحال عندما تكون إحدى العمليتين عملية قراءة؛ إذ إن القيمة المُسترجَعة من العنصر تتوقف على ترتـيـب عملية القراءة؛ من حيث كونـها قد تمَّت قبل عملية الكتابة أو بـعدها. فعلى سـبيل المثال: لو افترضنا وجود العنصر «س» (x) الذي يحتوي على القيمة «10» في قاعدة البيانات ووجود معاملتين (t1, t2)؛ فإن الترتيب التالي لعمليتَيْ قراءة العنصر (Read) من قِبل المعاملتين سينتج عنهما القيمة المُسترجَعة نفسها من العنصر، وهي «10» بغض النظر عن ترتيب العمليتين:

$$R_1[x]\ R_2[x] \equiv R_2[x]\ R_1[x]$$

أمَّا إذا افترضنا أن العملية التابعة للمعاملة الأولى؛ هي عملية كتابة (Write) على العنصر (x)؛ بحيث تصبح قيمته «5»، W1[x,5]؛ فإن ترتيب العمليتين مهمٌّ لكون القيمة المُسترجَعة من قِبل عملية القراءة التابعة للمعاملة الثانية، R2[x]، تختلف باختلاف ترتيبها في التنفيذ. وستكون القيمة المسترجعة «5» إذا نُفِّذت عملية القراءة بعد عملية الكتابة من قِبل المعاملة الأولى. أمَّا إذا نُفِّذت عملية القراءة قبل عملية الكتابة؛ فستكون القيمة المُسترجَعة «10». وإذا افترضنا أن كلتا العمليتين التابعتين للمعاملتين؛ هما: عمليات كتابة فإن ترتيبهما مهمٌّ أيضاً؛ إذ إن القيمة النهائية للعنصر تعتمدُ على القيمة التي قامت بكتابتها آخر معاملة. فعلى سبيل المثال: إذا افترضنا أن عملية المعاملة الثانية هي أيضاً عملية كتابة على العنصر (x) ولكن بقيمة «7»، W2[x,7]؛ فإن قيمة العنصر (x) ستكون «5» إذا كانت المعاملة الأولى هي آخر مَنْ قام بعملية الكتابة في حين ستكون قيمة العنصر «7» إذا كانت المعاملة الثانية هي آخر مَنْ قام بعملية الكتابة. ويعني هذا أن أيَّ عمليتين (Operations) تابعتين لمعاملتين مختلفتين على العنصر نفسه (x)، وليكونا (O1[x],O2[x])، تتضاربان إذا وُجِد بينهما عملية كتابة. ويُمثَّل مثل هذا التضارب، كما يلي:

$$O_1[x]\ O_2[x] \not\equiv O_2[x]\ O_1[x]$$

وباستخدام طريقة الأقفال؛ فإنه يجب على كلِّ عملية بسيطة تابعة لمعاملةٍ ما أن تتحصَّل على قفل على العنصر الذي ستتفاعل معه (سواءٌ من خلال قراءته أو بالكتابة عليه)، وفي الوَضْع الذي يتناسب مع طبيعة العملية. وعندما لا تستطيع العملية أن تتحصل على القفل بالوضع المناسب نتيجةً لوجود عملية أخرى تابعة لمعاملة أخرى قد سبق أن استحوذت على قفل على العنصر المطلوب، وفي وَضْعٍ يتضارب مع العملية المراد تنفيذها على العنصر؛ فإنه يتمُّ تأخير تنفيذ العملية حتى انتهاء المعاملة التي تمتلك القفل على العنصر. ويعني هذا أن المعاملة التي تتبعها التعليمة؛ يجب أن تنتظر لحين فكِّ القفل من قِبل المعاملات الأخرى التي تتضارب معها (في طبيعة القفل الذي وُضِعَ على العنصر) قبل أن تتمكَّن من مواصلة تنفيذ عمليتها. أمَّا إن كانت العملية الواجب تنفيذها لا تتضارب مع العملية (أو العمليات) التي سبق أن استحوذت على قفل على

العنصر (في حالة وجود مثل هذا القفل على العنصر)؛ فإنه يتمُّ وضع قفل على العنصر يبيِّن أن المعاملة التي تتبعها العملية قد استحوذت على قفل على العنصر، ومن ثم يتمُّ تنفيذ العملية. وعندما ينتهي تنفيذ كافة العمليات التابعة لمعاملةٍ ما (من خلال تنفيذ المعاملة لتعليمة التثبيت أو تعليمة الانسحاب)؛ يتمُّ فكُّ (أو تحرير) الأقفال كافةً التي تمَّ وضعها على العناصر التي تفاعلت معها المعاملة المنتهية. وبعد ذلك؛ يتمُّ السماح لأية معاملات أخرى متوقفة تنتظر انتهاء المعاملة من الحصول على الأقفال التي تنتظر الحصول عليها، ومن ثمَّ يتمُّ تنفيذ التعليمات التابعة لها. ويمكِن تمثيلُ تنفيذ تعليمات أيِّ معاملة، ولتكن (t1)، بنظام الأقفال، كما يلي:

$$rl_1[x]\ r_1[x]\ wl_1[y]\ w_1[y]\ rl_1[w]\ r_1[w]\ wl_1[z]\ w_1[z]\ c_1$$

ويعني التمثيلُ السابق للمعاملة (t1) أنه قد تمَّ تفكيكها إلى عمليات بسيطة تتكوَّن من عمليات قراءة وعمليات كتابة. كما يعني التمثيلُ أن كلَّ عملية بسيطة لا بد أن يسبقها عملية وَضْع قفل على العنصر الذي ستتفاعل معه العملية، وفي الوَضْع الذي يتناسب مع طبيعة العملية قبل تنفيذ العملية على العنصر. فعلى سبيل المثال: يُلاحَظ وَضْع قفل قراءة (Read LoCk (rl)) على العنصر (x) قبل تنفيذ تعليمة القراءة على العنصر. أمَّا بالنسبة للعنصر (y) فيُلاحَظ أنه قد تمَّ وَضْع قفل كتابة (Write LoCk (wl)) عليه قبل تنفيذ عملية الكتابة على العنصر (y)، مع تجاهل القيمة المفترَض كتابتها على العنصر في هذا المثال. ويُلاحَظ أيضاً أن المعاملة قد تمَّ إنهاء تنفيذها بتعليمه تثبيت (Commit (C)).

وباستخدام نظام الأقفال؛ فإن كلَّ معاملة تمرُّ في مرحلتين من مراحل تنفيذها، وهما: مرحلة النمو (Growing Phase)، ومرحلة الانحسار (أو الاضمحلال) (Shrinking Phase). ويتمُّ في مرحلة النموِّ وَضْع قفل على كلِّ عنصر تحاول المعاملة التعامل معه؛ وذلك قبل التعامل الفِعلي مع العنصر وفي وَضْع يتوافق مع طبيعة العملية. وبعد وَضْع القفل على العنصر يتمُّ التفاعل معه (أيْ: تنفيذ العملية المطلوبة عليه). أمَّا في مرحلة الاضمحلال؛ فيتمُّ فكُّ الأقفال التي تمَّ وضعها على العناصر التي قامت المعاملة بالتفاعل معها، ولا يحق للمعاملة التفاعل مع (أو وَضْع أيِّ أقفال إضافية على) أيِّ عناصر جديدة. ويتمُّ التعرُّف على مرحلة الاضمحلال من خلال تعليمة التثبيت أو تعليمة الإخفاق اللتين تمثلان انتهاء تعليمات المعاملات.

ويُعرَف نظام الأقفال السابق بنظام «الأقفال ذي المرحلتين» (Two-Phase LoCking (2PL)) لكون كلِّ معاملة لا بد أن تمرَّ بمرحلتين هما: مرحلة النموّ ومرحلة الاضمحلال، كما يضمن نظامُ الأقفال ذو المرحلتين تسلسلَ المعاملات، بحسب نظرية التسلسل، على الرغم من أن تنفيذ المعاملات يتمُّ بشكلٍ متزامن على قاعدة البيانات نفسها. ويُعَدُّ نظامُ الأقفال ذو المرحلتين الأكثر تطبيقاً في نظم إدارة قواعد البيانات؛ وذلك لسهولته وبساطته النسبية في البناء (أو التنفيذ) ضِمن نظم إدارة قواعد البيانات.

9–1–1–2 نظام الاستعادة (أو التشافي) (ReCovery ProtoCol):

يتمُّ بناءُ نظام الاستعادة (أو التشافي)؛ لضمان خاصية النووية (AtomiCity)، وخاصية الدوام (Durability). ويتمُّ ضمانُ خاصية النووية من خلال تأكيد أن أيَّ معاملة يجب تنفيذها باعتبارها وحدة منطقية متكاملة واحدة؛ فإما أن يتمَّ تنفيذ جميع عملياتها على العناصر التي تفاعلت معها المعاملة في قاعدة البيانات؛ وذلك عند انتهاء المعاملة ورغبتها في تثبيت عملياتها، وإمَّا ألا يتمَّ تنفيذُ أيٍّ من عملياتها على أيٍّ من العناصر التي تفاعلت معها المعاملة في حالة رغبة المعاملة في الانسحاب، أو في حالة وجود إخفاق (أو عطل في النظام). أمَّا خاصية الدوام؛ فيتمُّ ضمانها من خلال تأكيد أن أيَّ معاملة تمَّ الانتهاءُ من تنفيذ جميع عملياتها وتثبيت نتائجها على قاعدة البيانات سيستمرُّ وجود نتائجها على قاعدة البيانات بغض النظر عن أيِّ إخفاقات (أو أعطال) مستقبلية يتعرَّض لها النظام.

ومن أوسع الطرق انتشاراً في بناء نظم الاستعادة؛ هي تلك المبنيَّة على ما يُعرَف بسجلِّ الوقائع (Log or Journal). وسجل الوقائع عبارة عن ملف متسلسل (Sequential File or Append file) يُستخدَم من قِبل نظام إدارة قاعدة البيانات. ويحتوي سجلُّ الوقائع على بعض البيانات الأساسية للمعاملات، مثل: أرقامها، وبداياتها، ونهاياتها؛ بالإضافة إلى جميع التعديلات التي يتمُّ إجراؤها على قاعدة البيانات من قِبل المعاملات المختلفة. وكلما وَجَب إجراءُ تعديل على قاعدة البيانات؛ تتمُّ إضافة سجلات مناسبة وبشكلٍ متسلسل، الواحد تلو الآخر، على سجل الوقائع؛ بحيث تعكسُ طبيعة هذه التعديلات. ويعني هذا أن سجل الواقع يحتوي على تسلسل تصاعدي للتعديلات التي تمَّ إجراؤها على قاعدة البيانات. وتُعَدُّ سجلات الوقائع المبنيَّة على ما يُعرَف بالتدوين المُسبق للوقائع (Write-Ahead Logging)؛ هي أكثر سجلات الوقائع استخداماً من قِبل نظم

إدارة قواعد البـيانات. وكما يدلُّ مُسمَّى هذا النوع من سجلات الوقائع؛ فإن أيَّ تعديل على قاعدة البيانات يجب أن يُدوَّن في سجل الوقائع قبل أن يتمَّ إجراءُ أي تعديلٍ فعلي على قاعدة البـيانات. ويتمُّ ذلك من خلال تدوين تصويرتين لأيِّ عنصر قيد التعديل في سجل الوقائع، وهما: التصويرة السابقة (لعملية التعديل) (Before Image)، والتصويرة اللاحقة (لعملية التعديل) (After Image). فالتصويرة السابقة لعنصرٍ ما تمثِّل حالة (أو قيمة) العنصر قبل إجراء التعديل عليه من قِبل المعاملة، أمَّا التصويرة اللاحقة؛ فتمثل حالة (أو قيمة) العنصر بعد إجراء عملية التعديل عليه من قبـل المعاملة.

وتُستخدَم الصور السابقة لضمان خاصية النووية؛ وذلك عند انسحاب المعاملات من خلال تعليمات التراجع (أو الانسحاب) (Abort)، أو في حالة حدوث أعطال للنظام. فعند انسحاب معاملةٍ ما يجب، حسب خاصية النووية، إزالة كلِّ التعديلات التي أحدثتها المعاملة وانعكست فعلياً على قاعدة البـيانات. ويتمُّ ذلك من خلال استخدام الصور السابقة للعناصر التي تمَّ التعديل عليها من قبـل المعاملة. وكذلك هو الحال عند تعطل النظام؛ حيث تُستخدَم الصور السابقة؛ لضمان خاصية النووية، ولكن لإزالة كافة التعديلات التي قد أحدثتها المعاملات التي تمَّت مقاطعة تنفيذها؛ نتيجةً للعطل الذي أصاب النظام. وتُعَدُّ المعاملات التي تمَّت مقاطعتها غير منتهية التنفيذ لا من حيث تثبيت جميع نتائجها ولا من حيث انسحابها، وإلغاء جميع نتائجها من قاعدة البيانات؛ مما يعني عدم سلامة النتائج التي عكستها هذه المعاملات على قاعدة البـيانات؛ نتيجةً لمقاطعة تنفيذها الكامل حتى انتهائها. وتستدعي مقاطعة تنفيذ المعاملات إزالة أيِّ تأثير للتعديلات التي أحدثتها على قاعدة البيانات؛ من خلال استخدام الصور السابقة للعناصر التي تفاعلت معها هذه المعاملات.

أمَّا الصور اللاحقة؛ فتُستخدَم لضمان خاصية الدوام التي تتطلَّب استمرار نتائج المعاملات التي تمَّ تنفيذها، وانتهت بتثبيت نتائجها على قاعدة البيانات عند حدوث أعطال للنظام. فعند حدوث أيِّ عطل يتمُّ الرجوعُ لسجل الوقوعات؛ بهدف التعرُّف على المعاملات المنتهية بعمليات تثبـيت. ولكلِّ واحدة من هذه المعاملات تُستخدَم الصور اللاحقة المدوَّنة في سجل الوقوعات لتثبيت الصور اللاحقة للعناصر التي تمَّ التعديلُ عليها من قِبل المعاملة.

وباستخدام سجل التدوين المُسبق للوقائع؛ يتمُّ إجراءُ استرجاع قاعدة البيانات لحالة سليمة، بعد حدوث أية عطل للنظام، بشكلٍ يضمن كلاً من خاصية النووية وخاصية الدوام من خلال المرور

على سجل الوقوعات بثلاث مراحل، وهي كما يلي:

1- مرحلة التحليل (Analysis Phase): يتمُّ في هذه المرحلة المرورُ على سجل الوقوعات من البداية وحتى النهاية؛ وذلك للتعرُّف على جميع المعاملات المنفذة على قاعدة البيانات ووضعها التنفيذي من حيث انتهائها من خلال عمليات تثبيت أو عمليات انسحاب أو بشكلٍ غير طبيعي نتيجةً للعطل.

2- مرحلة إلغاء التعديلات (Undo Phase): يتمُّ لكلِّ معاملة منتهية بتعليمة إخفاق أو بشكلٍ غير طبيعي (لا يُوجَد لها تعليمة تثبيت، أو إخفاق في سجل الوقوعات)؛ استخدامُ الصور السابقة لإلغاء التعديلات التي أحدثتها المعاملة على قاعدة البيانات. وتتمُّ هذه المرحلة بشكلٍ عكسي على سجل الوقوعات؛ حيث يتمُّ إلغاء تعديلات المعاملات ابتداءً من نهاية السجل، بشكلٍ متسلسلٍ، وانتهاءً ببدايته. ويعني هذا أن عملية إلغاء التعديلات تتمُّ بشكلٍ يعاكس، من الناحية التاريخية (أو الزمنية)، التسلسل الزمني للتعديلات التي تمَّ إجراؤها على عناصر قاعدة البيانات من قِبل المعاملات المنسحبة أو المنتهية بشكلٍ غير طبيعي.

3- مرحلة استعادة التعديلات (Redo Phase): يتمُّ لكلِّ معاملة منتهية بتعليمة تثبيت إعادةُ تثبيت التعديلات التي أجرتها المعاملة على قاعدة البيانات باستخدام الصور اللاحقة للعناصر. وتتمُّ هذه المرحلة بشكلٍ تصاعديٍّ على سجل الوقوعات؛ وذلك من بدايته وحتى نهايته. ويعني هذا أن عملية استعادة التعديلات تتمُّ بشكلٍ يتوافق من الناحية التاريخية (أو الزمنية) مع التسلسل الزمني للتعديلات التي تمَّ إجراؤها على عناصر قاعدة البيانات من قِبل المعاملات التي تمَّ تثبيتُ نتائجها.

ومع مرور الزمن قد يصبح سجل الوقوعات طويلاً للغاية؛ بحيث تستغرق عملية الاستعادة وقتاً طويلاً جداً بعد حدوث أعطال للنظام. ليس هذا فحسب، وإنما قد يصبح سجل الوقوعات طويلاً جداً؛ بحيث يأخذ مساحةً تخزينيةً كبيرةً يتعذَّر معها تخزينه على القرص الصلب (أو الأقراص الصلبة) للحاسب الآلي. ولهذا السبب؛ فإنه من الضروري الحَدُّ من حجم سجل الوقوعات قدر ما أمكن ذلك. ومن الطرق المتبعة للحَدِّ من حجم سجل الوقوعات؛ ما يُعرَف بنقاط الاختبار (CheCkpoints). والفكرة الرئيسية وراء نقاط الاختبار؛ هي أن يقوم نظام إدارة قاعدة البيانات بوضع علامة معينة في سجل الوقوعات، تُسمَّى نقطة اختبار (CheCkpoint)، تدلُّ على

أن كافة المعاملات التي تقع قبل نقطة الاختبار قد تمَّ الانتهاء منها بشكلٍ كامل؛ سواءً من خلال تثبيت نتائجها على قاعدة البيانات، إذا كانت منتهيةً بعمليات تثبيت، أم من خلال إلغاء نتائجها من قاعدة البيانات، إذا كانت منتهية بتعليمات انسحاب أو كانت مخفقةً لسبب ما. ونظراً لأن هذه المعاملات تُعَدُّ منتهيةً بالكامل؛ فإنه لا يجب على نظام إدارة قاعدة البيانات النظر في آثارها على قاعدة البيانات عند حدوث عطل للنظام. أمَّا بالنسبة للمعاملات التي لها سجلات، في سجل الوقوعات، تقع بعد نقطة الاختبار؛ فإنه يتوجَّب النظر فيها في أثناء عملية الاستعادة. وبهذه الطريقة يُمكِن الحَدُّ من عدد المعاملات التي يجب العمل على استعادتها بعد حدوث أيِّ عطل للنظام كما يُمكِن تقليصُ حجم سجل الوقوعات من خلال الاستغناء عن المساحة التخزينية المخصَّصة لسجلات الوقوعات التي تقع قبل نقاط الاختبار من خلال عملية تُسمَّى عملية جمع النفايات (Garbage ColleCtion). وتختلف نظم إدارة قواعد البيانات في الطرق التي تتبعها؛ لوَضْع نقاط الاختبار والتعامل معها؛ ولكنها تتفق جميعاً على المفهوم الرئيسي لنقاط الاختبار.

9–1–2 الزنادات والإجراءات المتكرِّرة (Triggers and Routines):

لم يكن من ضِمن لغة الاستفسار البنائية قبل مقياس (SQL-99) دعم للإجراءات (ProCedures)، أو الدوال (FunCtions) التي من الممكن أن يقومَ المستفيدون بتعريفها؛ وذلك على الرغم من أن نظم قواعد البيانات المتوفرة على المستوى التجاري؛ توفِّر إمكانية تعريف مثل هذه الإجراءات والدوال. وما الزنادات (Triggers) إلا إجراءات (Routines) ذات مُسمَّيات. ويتكوَّن الزناد الواحد من مجموعةٍ من تعليمات لغة الاستفسار البنائية. وتقع الزنادات تحت تحكُّم نظام إدارة قواعد البيانات؛ بحيث يتمُّ تنفيذ التعليمات التي يتكوَّن منها الزناد عندما تتحقق الشروط التي يتطلبها تنفيذ الزناد. ويرتبط كلُّ زناد بحدث معين يؤدي إلى تنفيذه، وهذه الأحداث هي تعليمات التعديل (الإضافة والحذف والتحديث). ولأن هذه التعليمات تقوم بتغيير محتوى قاعدة البيانات، ومن ثم حالة قاعدة البيانات؛ فإن الشروط المرتبطة بالزنادات قد تتحقق عند التعديل على قاعدة البيانات، ثم يقوم نظام إدارة قاعدة البيانات بتنفيذ الزنادات التي يتمُّ تحقق شروط تنفيذها.

وتُعَدُّ الزنادات أحد مكامن القوة في نظم إدارة قواعد البيانات؛ حيث يتمُّ كتابتها مرة واحدة، ومن ثم يتمُّ التحكُّم في تنفيذها من خلال نظام إدارة قاعدة البيانات. وبهذه الطريقة يتمُّ التحكُّم في فرض تكامل البيانات وتناسقها بشكلٍ أكبر، هذا بالإضافة إلى تقليص عدد التعليمات التي يجب

كتابتها، بشكلٍ متكرر، من قبل التطبيقات المختلفة لتطبيق محتويات الزناد نفسها. ويتكون كلٌّ من الزنادات والإجراءات من مجموعةٍ من التعليمات الإجرائية؛ غير أن تنفيذ الزنادات يتمُّ تلقائياً من قِبل نظام إدارة قاعدة البيانات، في حين لا يتم تنفيذ الإجراءات إلا بعد عملية مناداتها من قِبل نظم التطبيقات.

9–1–2–1 الزنادات:

يتمُّ تنفيذ الزنادات من قِبل نظام إدارة قاعدة البيانات عندما تتحقق شروطها بغض النظر عن التطبيق الذي أدَّى إلى تحقق شروط تنفيذها. ومن الممكن أن يتسلسل تنفيذ الزنادات؛ بحيث يؤدي تنفيذ أحد الزنادات إلى تنفيذ زناد آخر، وهكذا. وقد يتمُّ استخدام الزنادات، على سبيل المثال؛ للتحقق من قيود السلامة المرجعية أو لفرض بعض قواعد العمل المعمول بها في المنظمة أو لتفعيل إجراء معين. وتتكوَّن الزنادات من ثلاثة مكوِّنات رئيسية، وهي:

- الحدث (EVENT): يمثل التغيرُ في حالة قاعدة البيانات الذي يؤدي إلى تفعيل (أو تنشيط) الزناد.

- الشرط (CONDITION): الاختبار أو الاستفسار الذي يجب التحقق منه عند تفعيل الزناد. وعندما يكون الشرط استفساراً (Query)؛ فإن الشرط يُعَدُّ متحققاً (True) عندما تكون نتيجة الاستفسار غير خالية، أيْ: يُوجَد ناتجٌ للاستفسار.

- الفعل (ACTION): مجموعة التعليمات التي سيتمُّ تنفيذها عندما يتم تفعيل الزناد ويتحقق الشرط المصاحب له.

والشكل (9-1) يوضِّح الشكل العام لتعريف الزنادات.

```
CREATE [OR REPLACE] TRIGGER trigger_name
      {BEFORE | AFTER} {INSERT | DELETE | UPDATE} ON
Table_name
      [FOR EACH ROW [WHEN (Trigger_Condition)]]
      Trigger_body;
```

شكل رقم (9–1): الشكل العام لتعريف الزنادات.

ومن الأمورِ التي يتم مراعاتها عند تعريف الزنادات؛ هو التوقيت الذي سيقوم بتفعيل الزناد؛ بحيث إنه قد يتمُّ تفعيل الزناد إما قبل أو بعد التعليمة المفترض أن تقوم بتفعيله، وقد تكون إمَّا تعليمة إضافة أو حذف أو تحديث، وعمَّا إذا كان سيتمُّ تفعيلُ الزناد مرة واحدة على مستوى التعليمة (بشكل كامل) أو مرةً واحدةً على مستوى كل صف
(أو سجل) يتأثر بسبب تنفيذ التعليمة.

9–1–2–2 الإجراءات المتكررة (Routines):

من الممكن أن تكون الإجراءات المتكررة في (SQL) على هيئة إجراءات
(ProCedures) أو دوال (FunCtions). ويدخل لكلِّ دالة قيمة (Parameter) واحدة، وتعيد قيمة واحدة كذلك. أمَّا الإجراءُ؛ فيمكن أن يكون له قِيم مُدخَلة، أو قِيم مُستعَادة، أو كلا الاثنين معاً. والشكل (9-2) يوضِّح الشكل العام لتعريف الإجراءات المتكررة.

```
{CREATE PROCEDURE | CREATE FUNCTION} routine_name
        ({parameter} [{, parameter} ...]})
        [RETURNS data_type result_cast]                 /* for functions only*/
        [LANGUAGE {ADA | C | COBOL | FORTRAN | MUMPS | PASCAL | PL1 | SQL}]
        [PARAMETER STYLE {SQL | GENERAL}]
        [SPECIFIC specific_name]
        [DETERMINISTIC | NOT DETERMINISTIC]
        [NO SQL | CONTAINS SQL | READS SQL DATA | MODIFIES SQL DATA]
        [RETURN NULL ON NULL INPUT | CALL ON NULL INPUT]
        [DYNAMIC RESULT SETS unsigned_integer]          /* for procedures only */
        [STATIC DISPATCH]                               /* for functions only */
        routine_body;
```

شكل رقم (9–2): الشكل العام لتعريف الإجراءات المتكررة.

9–2 قواعد البيانات الشيئية (ObjeCt–Oriented Database Systems):

تَسدُّ نماذج البيانات التقليدية، وخاصةً النموذج العلاقي؛ الاحتياجات التقنية التي يتطلبها تطويرُ نظم التطبيقات المتعلقة بمكننة أعمال المنظمات ذات الصبغة التقليدية. غير أن هذه النماذج تعاني بعض القصور عند محاولة تصميم وتنفيذ نظم تطبيقات أكثر تعقيداً، مثل: تطبيقات التصميم بمساعدة الحاسب الآلي (Computer-Aided Design)، والتصنيع بمساعدة الحاسب الآلي

(Computer-Aided ManufaCturing)، والتجارب العلمية، ونظم المعلومات الجغرافية (GeographiCal Information Systems)، وتطبيقات الوسائط المتعددة (Multimedia AppliCations)؛ على سبيل المثال لا الحصر. والسبب وراء هذا القصور يَرْجِع إلى أن خصائص ومتطلبات هذه التطبيقات تختلف عن خصائص ومتطلبات التطبيقات التقليدية، مثل احتوائها على عناصر (أو أشياء (ObjeCts)) أكثر تعقيداً، وتنفيذها لمعاملات أكثر طولاً من حيث الفترات الزمنية التي تحتاج إليها للتنفيذ، وحاجتها إلى أنواع بيانات جديدة لحِفظ الصور والأصوات والبيانات النصية الطويلة بالإضافة إلى حاجتها إلى إجراء عمليات غير تقليدية على أنواع البيانات الجديدة.

وقد تمَّ اقتراحُ النموذج الشيئي؛ لتلبية احتياجات التطبيقات الأكثر تعقيداً، مثل تلك المذكورة أعلاه. ويوفر النموذج الشيئي المرونة؛ من حيث عدم التقيُّد بنوعية بيانات معينة أو تعليمات مُحدَّدة للتعامل مع محتويات قاعدة البيانات مقارنةً بنظم قواعد البيانات التقليدية. ومن المزايا الرئيسية للنموذج الشيئي تمكينُ مُصمِّمي قاعدة البيانات من توصيف هياكل الأشياء التي يرغبون في نمذجتها؛ بالإضافة إلى العمليات التي يمكِن إجراؤها عليها. ومن الأمور المهمة الأخرى وراء اقتراح النموذج الشيئي في نمذجة قواعد البيانات كون غالبية لغات البرمجة الحديثة تعتمدُ على المفاهيم الشيئية؛ مما يشكِّل صعوبةً في تطوير نظم التطبيقات ما لم يكن لنظام قاعدة البيانات القدرة على التعامل مع هذه اللغات بشكلٍ مباشر، ومن خلال استخدامها للمفاهيم ذاتها. لذلك؛ فإننا نجد أن قواعد البيانات الشيئية تتبنَّى العديدَ من المفاهيم التي تمَّ تطويرها أساساً للغات البرمجة الشيئية. وتمثل المفاهيم الأساسية للنموذج الشيئي محور هذا الجزء من الكتاب.

9–2–1 مفاهيم الأشياء الموجهة (**ObjeCt–Oriented ConCepts**):

إن أصلَ مصطلح الأشياء الموجهة (ObjeCt-Oriented) يعود إلى لغات البرمجة (Elmasri and Navathe, 2015). غير أن المفاهيم الرئيسية وراء هذا المصطلح نراها اليوم مطبقةً ليس في لغات البرمجة فحسب، وإنما في نظم قواعد البيانات، وهندسة البرمجيات (Software Engineering)، وقواعد المعرفة (Knowledge Base)، ونظم الحاسب الآلي بشكلٍ عام. وكان من أول المفاهيم التي تمَّ تطبيقها في الأشياء الموجّهة مفهوم الصنف (Class)

الذي يقوم بتجميع الأشياء المتشابهة ضِمن هيكل واحدٍ يُسمَّى الصنف. وبناءً على ذلك ظهر مفهوم أنواع البيانات المجرَّدة (AbstraCt Data Types) الذي يُقصَد منه إخفاء الهياكل الداخلية الخاصة بالأشياء التي تمَّ تعريفها عن المستفيدين، وفي الوقت نفسه توفير عمليات تمكِّنهم من التعامل مع هذه الأشياء. وأدَّى هذا إلى ظهور مفهوم التغليف (EnCapsulation) الذي يُقصَد منه إخفاء المعلومات عن المستفيدين. ويمكِن توضيحُ معنى هذا المفهوم إذا تصوَّرنا أن الأعداد الصحيحة (Integers) (أو الحقيقية (Real)) عبارة عن أشياء لها هياكل، وأنه يمكن إجراءُ عمليات مُعيَّنة عليها (مثل: الجمع والطرح). في هذه الحالة تكون الطريقة التي يتمُّ تخزين هياكل هذه الأعداد عليها مُخفاة عنا (كمستفيدين) ولا نعلم عنها شيئاً؛ ولكننا نستطيعُ تعريف الأعداد ضِمن البرامج التي نقوم بكتابتها والتعامل مع هذه الأعداد من خلال العمليات التي يوفرها لنا نظام الحاسب الآلي. وهذا المثال شبيه بما يُقصَد به مفهوم التغليف؛ غير أنَّ الأشياء التي يحاول أن يخفيها هذا المفهوم تكون عادةً ذات هياكل بيانات أكثر تعقيداً. وقد تمَّ لاحقاً ظهورُ مفاهيم إضافية للأشياء الموجهة من ضِمنها تمريرُ الرسائل (Message Passing) والتوريث (InheritanCe). وفيما يلي استعراض لأهمِّ مفاهيم النموذج الشيئي.

9-2-1-1 مفهوم الشيء:

يتكوَّن الشيء (ObjeCt)، سواءٌ في لغات البرمجة أو في نظم قواعد البيانات الشيئية؛ من مكوِّنين رئيسيين، هما: حالة (أو قيمة) الشيء (State (or Value))، وسلوكه (أو عمله) (Behavior (or Operation)). وعلى الرغم من أن الأشياء قد تختفي بعد تنفيذ البرمجيات (في لغات البرمجة)؛ فإن الأشياء في نظم قواعد البيانات الشيئية تتصف بالدوام (Durability)؛ بحيث يبقى وجودها ضِمن قاعدة البيانات حتى بعد انتهاء تنفيذ المعاملات التي تتعامل معها (ما لم تتم إزالتها بشكلٍ صريح). ويعني هذا أن قواعد البيانات الشيئية تقوم بتخزين الأشياء في الذاكرة الثانوية بشكلٍ دائم، وتسمح بأن تتم مشاركة التعامل مع هذه الأشياء من قِبل البرامج والتطبيقات المختلفة. وتتطلَّب عملية المشاركة للأشياء ووجودها الدائم، وعلى خلاف الأشياء في لغات البرمجة، استخدام التقنيات الخاصة بنظم قواعد البيانات، مثل: الفهرسة، ونظم التحكُّم في التزامن،

ونظم الاستعادة (أو التشافي). وحتى يمكِن التمييزُ بين الأشياء المختلفة في قاعدة البيانات؛ فإن لكل شيء ذاتيته الخاصة به التي تميِّزه، وبشكلٍ منفرد، عن بقية الأشياء في قاعدة البيانات.

9-2-1-1-1 ذاتية الشيء (ObjeCt Identity):

لكلِّ شيءٍ يُخزَّن في قاعدة بيانات شيئية ذاتيتُه الخاصة به التي يقوم نظام إدارة قواعد البيانات الشيئية بإسنادها له. وتتمثل ذاتية الشيء بمعرِّف خاصٍّ به (ObjeCt Identifier) يتمُّ توليده وإسناده إلى الشيء من قِبل النظام. ولا تكون قيمة المعرِّف الخاصة بالشيء ظاهرة للمستخدمين، وإنما تكون خاصةً بالنظام؛ حتى يتمَكَّن من التمييز بين الأشياء المختلفة المخزَّنة في قاعدة البيانات، وحتى يتمَكَّن من إدارة الارتباطات (أو العلاقات) بين الأشياء المختلفة. وأهمُّ خاصية لذاتية الشيء؛ هي عدم تغيُّرها بمرور الزمن. لذلك؛ فإنه يجب في نظم قواعد البيانات الشيئية أن تحتوي على آليات مناسبة تمَكِّن من توليد ذاتية خاصة لكلِّ شيء يُخزَّن في قاعدة البيانات. أمَّا الخاصية الثانية لذاتية الشيء؛ فهي عدم استخدامها أكثر من مرة؛ بمعنى أنه عندما تتم إزالة شيء معين من قاعدة البيانات؛ فإنه يجب عدم إعادة استخدام ذاتيته مع شيء آخر. والسبب وراء ذلك منطقي؛ إذ إن كل شيء في الطبيعة له ذاتيته الخاصة التي لا يمكن إعادة استخدامها مرةً أخرى للدلالة على أيِّ شيء آخر.

وتعني الخاصيتان أعلاه أن ذاتية الشيء يجب ألا تعتمد على قِيم خصائصه (Attributes)؛ لأن قِيم خصائص الشيء قد تتغير بمرور الزمن، كما أن ذاتية الشيء يجب ألا تعتمد على عنوان موقع (أو مكان) تخزين الشيء في الذاكرة الثانوية للحاسب الآلي؛ لأن عنوان الموقع قد يتغير مع مرور الزمن نتيجةً لإعادة ترتيب الأشياء في الذاكرة الثانوية للحاسب الآلي، ومن ثم إسنادها لأشياء أخرى في قاعدة البيانات. ويعني هذا أن قواعد البيانات الشيئية تقوم بتوليد وإسناد ذاتية فريدة لكلِّ شيء يُخزَّن في قاعدة البيانات. وبمقارنة ذاتية الشيء والمفتاح الرئيسي للعلاقات (أو الجداول) في النموذج العلاقي؛ نجد أنه عندما تتغير قيمة المفتاح الرئيسي لسجلٍّ ما، في النموذج العلاقي؛ فإن ذاتيته تتغير على الرغم من أنه ما زال يمثل الشيء نفسه على أرض الواقع. بالإضافة إلى ذلك؛ فإنه قد يكون للمفتاح الرئيسي للشيء أكثر من مُسمًّى في علاقات قاعدة البيانات؛ مما يصعب معه الجزم بأن الشيء هو ذاته في العلاقات المختلفة. فعلى سبيل المثال: قد يكون المفتاح الرئيسي في علاقة ما هو «رقم الموظف» (Emp_No)؛ في حين يكون في علاقةٍ أخرى هو «رقم

السجل المدني» (Emp_SSN)، اللذان يمثلان على أرض الواقع الخاصية نفسها. لذلك؛ فإن نظم البيانات الشيئية تتغلَّب على هاتين المعضلتين من خلال إسنادها إلى ذاتية خاصة لكلِّ شيء يُعرَف في قاعدة البيانات وفق الخاصيتين المذكورتين أعلاه.

9–2–1–1–2 حالة (أو قيمة) الشيء (State (or Value) of an ObjeCt):

حالة الشيء هي القِيم الفعلية لخصائصه وللعلاقات التي تربطه مع الأشياء الأخرى في قاعدة البيانات في لحظةٍ زمنية ما. ويعني هذا أن حالة الشيء تتغير من وقتٍ لآخر في أثناء فترة حياته (أو تواجده). ويتمُّ تحديد حالة الشيء في وقتٍ ما من خلال القِيم التي تحتويها خصائصه؛ بالإضافة إلى ارتباطاته مع الأشياء الأخرى في قاعدة البيانات. وتتغير حالة الشيء من وَضْعٍ إلى آخر حسب العمليات التي تُنفَّذ عليه وتغيِّر من سلوكه (أو عمله).

9–2–1–1–3 سلوك (أو عمل) الشيء (Behavior (or Operation) of ObjeCts):

سلوك الشيء يمثل تفاعله مع العالم الخارجي؛ من خلال العمليات التي تُنفَّذ عليه. ويعتمد سلوك الشيء على الحالة التي يُوجَد عليها وعلى طبيعة العملية المنفذة عليه. أمَّا العملية؛ فما هي إلا فعلٌ يتمُّ على الشيء ويؤثر على حالته وإرجاع نتيجة الفعل للمستفيد (أو التطبيق أو الشيء) الذي قام باستدعاء العملية.

فعلى سبيل المثال: لنفترض وجود الطالب «أحمد صالح» ممثلاً كشيء ضِمن قاعدة بيانات شيئية. في هذه الحالة؛ يكون للشيء الذي يمثل الطالب «أحمد صالح» ذاتيته التي تميِّزه عن بقية الأشياء في قاعدة البيانات، وفيهم الطلبة الآخرون، ومجموعة من الخصائص التي تميِّز الطالب، مثل: اسمه، وعنوانه، وتخصُّصه، وما إلى ذلك من خصائص أخرى تتعلق بالطلبة. وتتمثل حالة الشيء «أحمد صالح» في القِيم الفعلية لخصائصه في لحظةٍ ما بالإضافة إلى العلاقات التي تربطه بالأشياء الأخرى في قاعدة البيانات. أمَّا سلوك الشيء «أحمد صالح»؛ فيتمثل من خلال تفاعله مع العمليات التي تُجرَى عليه مثل حساب «المعدل التراكمي»، أو حساب «العمر». وبناءً على هذا؛ فإن الشيء «أحمد صالح» عبارة عن حزمة (PaCkage) تتكوَّن من حالة الشيء (أيْ: قِيم خصائصه وارتباطاته) وسلوكه (أيْ: تفاعله تجاه العمليات التي تُنفَّذ عليه).

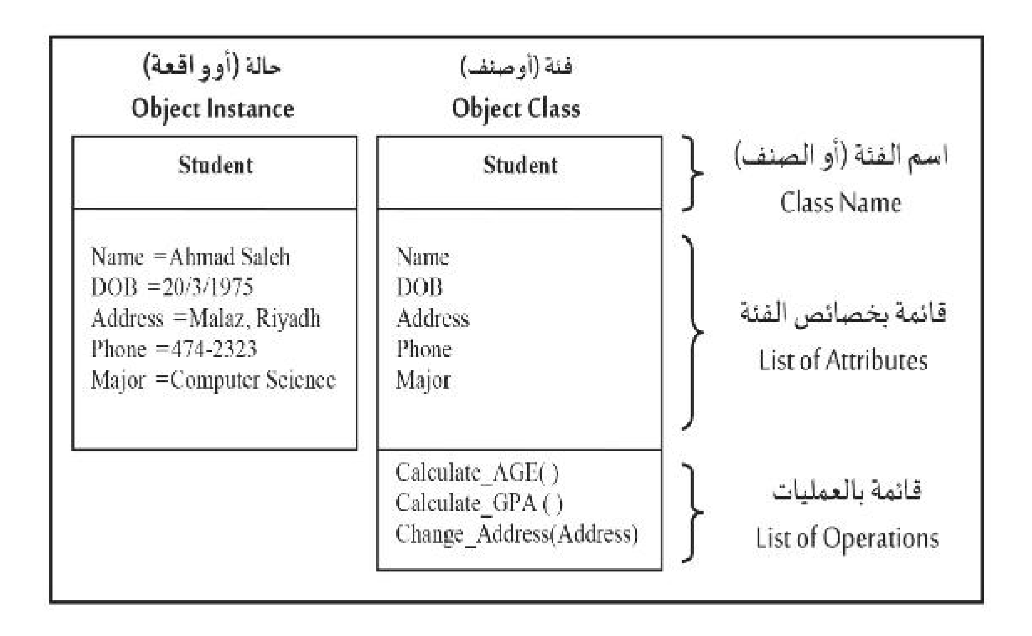
9–2–1–2 الفئة (أو الصنف) (Class):

عند حديثنا عن نموذج «كينونة - علاقة»؛ تمَّ التفريق بين الكينونة وفئة الكينونة؛ وذلك لإزالة أيِّ التباس بين المقصود بحالة (أو واقعة) معينة من الحالات (أو الوقوعات)؛ وفئة الكينونة التي تتبعها الحالة (أو الواقعة). وكذلك هو الحال في قواعد البيانات الشيئية؛ إذ يتمُّ التفريق بين حالة (أو واقعة) من حالات (أو وقائع) الشيء (ObjeCt InstanCe)؛ والصنف (ObjeCt Class) الذي تتبعه الحالة (أو الواقعة). ويعني هذا أن الحالة الواحدة من حالات الشيء؛ لها وجودها في وقتٍ ما، وقد تحتلُّ مساحةً مُحدَّدةً في الطبيعة؛ غير أن الصنف عبارة عن شيء مجرد (AbstraCt) يُعبر عن مجموعة من الأشياء التي تشترك في مخططات هياكلها (StruCture) وسلوكها (أو تفاعلها) (Behavior) مع ما حولها. وقد يتساءل المرءُ عن الفرق بين الكينونة، كما تمَّ تعريفها في نموذج «كينونة - علاقة»، والشيء، كما تمَّ تعريفه في النموذج الشيئي. والواضح أنه بالإمكان تمثيل كلِّ كينونة في نموذج «كينونة - علاقة» على أنها شيءٌ في النموذج الشيئي. غير أنه بالإضافة إلى تخزين المعلومات المتعلقة بحالة الكينونة (التي تتمثل في قِيم خصائصها والعلاقات التي تربطها مع الكينونات الأخرى)؛ فإن للشيء سلوكاً يتمثل في عمليات من الممكن أن تُنفَّذ عليه بغية معرفة الحالة التي هو عليها أو للتغيير فيها.

ويمثل الشكل رقم (9-3) فئة (أو صنفاً) من الأشياء وهم الطلبة، وحالة (أو واقعة) من وقوعات هذه الفئة (أو الصنف)؛ وهي الطالب «أحمد صالح». ويظهر في أعلى الشكل، الممثل للصنف، اسم الصنف تتبعه قائمة بالخصائص المتعلقة به (كما تظهر في الجزء الأوسط من الشكل). أمَّا في أسفل الشكل؛ فتظهر قائمة بالعمليات التي من الممكن تنفيذها على هذه الفئة (أو الصنف) من الأشياء.

ويمثِّل صنف الطلبة، في هذا المثال، مجموعة حالات الطلبة التي لها مخطط هيكل مشترك وسلوك مشترك أيضاً. فكلُّ حالات الطلبة تشترك في خصائص «الاسم» و«تاريخ الميلاد» و«العنوان» و«رقم الهاتف» و«التخصص». كما أن كلَّ حالات الطلبة لها السلوك نفسه من خلال العمليات المشتركة التي بالإمكان أن تُنفَّذ عليها، وهي: «حساب عمر الطالب» (CalCulate_AGE( ))، و«حساب المعدل التراكمي» (CalCulate_GPA( ))، و«تغيير العنوان» (Change_Address(Address)). وبذلك؛ فإن كلَّ صنفٍ عبارة عن نموذج (أو مخطط) للحالات التي يمثلها. وكلُّ حالة تَعْرِف الصنف الذي تتبعه. فحالة الطالب «أحمد

صالح» تَعْرِف أنها تتبع لصنف «الطلبة». كما أن الحالات التي تتبع لصنفٍ ما من الممكن أن تشارك في علاقات متشابهة مع أشياء أخرى. فعلى سبيل المثال: كلُّ الطلبة يقومون «بالتسجيل» في مواد دراسية. لذلك؛ فإنه من الممكن أن يرتبط صنف الطلبة بعلاقة تُسمَّى «تسجيل في» (Register_for) مع صنفٍ آخر يُسمَّى «المادة الدراسية» (Course).



شكل رقم (9–3): فئة (أو صنف) الطلبة وحالة (أو واقعة) منها.

9–2–1–2–1 أنواع العمليات:

يتمُّ تحديدُ حالة الشيء من خلال قِيم خصائصه وارتباطاته مع الأشياء الأخرى في قاعدة البيانات. أمَّا سلوك الشيء؛ فيعتمد على حالته وعلى طبيعة العملية المنفذة عليه. والعملية عبارة عن استدعاء (InvoCation) لفعل (ACtion) يقوم به شيء ما في النظام على شيء آخر بغية الحصول على رَدَّة فعل (Response) من الشيء. ويمكِن تصوُّر عمليةٍ ما على أنها خدمة معينة يقوم بتوفيرها شيءٌ ما لعملائه. وعند استدعاء العميل للعملية بغية الحصول على الخدمة؛ فإنه يقوم بإرسال رسالة إلى الشيء مقدِّم الخدمة، تبيِّن طبيعة الخدمة المطلوبة. وبمجرد استلام طلب الخدمة؛ يقوم مزوِّد الخدمة بتنفيذ العملية وإعادة نتيجتها للعميل الذي قام بطلبها (أو استدعائها).

ويمكِن تصنيفُ العمليات التي بالإمكان استدعاؤها (أو تنفيذها) على الأشياء المخزَّنة في قواعد البيانات الشيئية إلى أربعة أصناف، وهي كما يلي:

1- عمليات إنشاء (ConstruCtor Operations): يقوم هذا النوعُ من العمليات بـإنشاء حالة (أو واقعة) جديدة من حالات الفئة (أو الصنف). فعلى سبيل المثال: يمكِن إنشاء حالة جديدة من حالات الطلبة، كما يلي:

```
Create Student("Ahmed Saleh", 20/3/1975, "Malaz, Riyadh", "Computer Science")
```

ولتوافر هذا النوع من العمليات لجميع فئات (أو أصناف) الأشياء؛ فإنه لا يتم ذكره بشكلٍ ظاهر (ExpliCit) عند تعريف فئات الأشياء.

2- عمليات استفسار (أو استرجاع) (Query (or SeleCtor) Operations): يقوم هذا النوع من العمليات بـاسترجاع حالة الشيء دون إحداث أيِّ تغييرات عليها. فعلى سبيل المثال: يمكِن استرجاع عنوان طالب معين باستخدام التعليمة (Get_Address( )). وهذه العملية غير مُوضَّحة في الشكل رقم (9-3)؛ لكون مثل هذه العملية ليس من الضروري إدراجها ضمن عمليات الصنف؛ لأنها تسترجع قيمة خاصية أساسية من خصائص الصنف. أمَّا عملية «حساب عمر الطالب» (CalCulate_AGE( ))؛ فهي من العمليات التي يجب إظهارها ضِمن عمليات الصنف على الرغم من كونها عملية استرجاع لا تؤثر في حالة الطالب الذي تقوم العملية بحساب عمره. والسبب وراء إظهار هذه العملية ضِمن عمليات الصنف يرجعُ إلى أن هذه العملية لا بد أن ترتبط بطريقة (Method) تبيّن بناء آلية حساب عمر الطالب. وعادةً تُبنَى هذه العملية على أساس أنها اشتقاق من خاصية تاريخ الميلاد؛ بحيث يتم طرح تاريخ ميلاد الطالب من تاريخ اليوم الذي تم فيه استدعاء العملية. وتاريخ اليوم هو من القيم المتغيرة التي يمكن استرجاعها من نظام الحاسب الآلي وتُسمَّى عادةً (SYS_DATE).

3- عمليات تعديل (ModifiCation Operations): يقوم هذا النوع من العمليات بـتغيير حالة الشيء. فعلى سبيل المثال: عندما يتم استدعاء العملية (Change_Address(Address))؛ فإن هذه العملية ستقوم بتغيير القيمة المخزَّنة في حقل عنوان الطالب؛ ليصبح مساوياً للقيمة المُدرَجة ضمن محددات (Arguments) العملية (وهي القيمة المخزَّنة في (Address)). ويتمُّ إيضاح

المحددات الظاهرة (ExpliCit) لأيِّ عملية ضمن قوسين. أمَّا في حالة عدم وجود محددات ظاهرة؛ فتترك العملية في قوسين فارغين.

4- عمليات إتلاف (DestruCtion Operations): إن عمليات الإتلاف، وعلى النقيض من عمليات الإنشاء؛ تقوم بإزالة حالة معينة من حالات الصنف من قاعدة البيانات. فعلى سبيل المثال: يمكِن استخدام العملية (Destroy_Student( )) لإزالة حالات الطلبة. ولأن هذا النوع من العمليات متوفر في جميع فئات (أو أصناف) الأشياء، وكما هو الحال في عمليات الإنشاء؛ فإنه لا يتم ذكره بشكلٍ ظاهر (ExpliCit) عند تعريف فئات الأشياء.

كما يُوجَد هناك صنف خامس من العمليات؛ ولكن هذا الصنف لا يُطبق على (حالات أو وقوعات) الأشياء فردياً؛ بل تُطبق على الصنف الذي تتبعه مجموعة من الأشياء تطبيقاً جماعياً. ويُسمَّى هذا الصنف من العمليات «عمليات مساعدة» (Class Utilities) أو «عمليات أصناف» (Class Operations). فعلى سبيل المثال: تُعَدُّ عملية «حساب عدد الطلاب في تخصص» (Calulate_No_of_Students (Major)) من هذا النوع من العمليات؛ لأنه يجب تطبيقها على جميع الحالات المتوافرة من صنف الطلبة؛ لمعرفة العدد الكلي للطلبة الدارسين في تخصُّص معين.

9-2-1-3 مفهوم التغليف (EnCapsulation):

يُعَدُّ مفهوم التغليف أحد المفاهيم المميزة للنظم واللغات الشيئية؛ إذ إن هذا المفهوم غير مطبق في نظم ونماذج قواعد البيانات التقليدية. فمن المعتاد عليه في النظم والنماذج التقليدية أن يكون هيكل مكونات قاعدة البيانات ظاهراً للمستفيدين وبرامج التطبيقات. فعلى سبيل المثال: يتوافر في نموذج البيانات العلاقي عمليات الاختيار والإضافة والحذف والتحديث التي يمكِن تطبيقها على أيِّ علاقة (أو جدول) في قاعدة البيانات. ويعني هذا أن هذه العمليات ذات صبغة عامة، غير مُخصَّصة لشيء (ObjeCt) مُحدَّد من الأشياء المخزَّنة في قاعدة البيانات (العلاقية). كما أن العلاقات وحقولها تُعَدُّ ظاهرةً للمستفيدين وبرامج التطبيقات؛ بحيث يمكِن التعامل معها بشكلٍ مباشر من خلال التعليمات العامة التي يوفرها النموذج.

ويرتبط مفهوم التغليف بمفهوم «أنواع البيانات التجريدية» (AbstraCt Data Types) ومفهوم «إخفاء البيانات» (Information Hiding) المعروفين في لغات البرمجة الموجهة

للأشياء. ويُقصَد بهذه المفاهيم، عند الحديث عن نظم قواعد البيانات الشيئية، تعريف سلوك فئة الشيء من خلال العمليات التي من الممكن أن تُستدَعى وتُنفَّذ على الشيء؛ بحيث يكون هيكلُ (أو مكونات) الشيء مخفياً عن العالم الخارجي، وبشكلٍ لا يُسمَح بـالتعامل معه إلا من خلال مجموعة من العمليات المحدَّدة التي توفرها فئة (أو صنف) الشيء. وبهذه الطريقة يكون المستفيدون على علم بالواجهة التي يقدّمها الشيء للتعامل مع محتواه، والتي تتمثل في اسم كلِّ عملية يمكِن تنفيذها على الشيء والمحَدِّدات (أو المتغيرات) (Parameters (or Arguments)) التي تتطلبها كلُّ عملية. أمَّا طريقة بناء الشيء والكيفية التي يقوم فيها بتنفيذ العمليات، بما في ذلك هيكله (Data StruCture) وبناء عملياته (Operation’s Implementation)؛ فتكون مخفيةً عن المستفيدين. ويمكن تشبيه العملية الواحدة بالإجراء (ProCedure) (أو الدالة (FunCtion)) في لغات البرمجة؛ حيث يتمُّ استدعاء الإجراء (أو الدالة)؛ من خلال مُسمَّاها بـالإضافة إلى قِيم مُحدِّداتها (أو متغيراتها). أمَّا طريقة عمل الدالة أو هياكل الـبيانات التي تحتويها؛ فتكون مخفاةً عن المستفيدين من الإجراء (أو الدالة).

وتُسمَّى واجهة العملية التي يتمُّ تعريفها في فئة الصنف «توقيعاً» (Signature). أما الـبناء الفعلي لطريقة تنفيذ العملية داخل فئة الصنف؛ فيُسمَّى «طريقة» (Method). ويتمُّ استدعاء إحدى الطرق المعرفة لشيءٍ ما من خلال إرسال رسالة (Message) للشيء. وبعد ذلك؛ يقوم الشيء بتنفيذ الطريقة المخصصة لتنفيذ العملية المطلوبة وإعادة نتيجة التنفيذ للشيء (أو المستفيد) الذي قام بـإرسال الرسالة. وقد يتمُّ في أثناء تنفيذ طريقة ما إرسالُ رسالة من شيء إلى شيء آخر بـشكلٍ متداخل.

4-1-2-9 التحميل الزائد (**Polymorphism**):

إن أصل «التحميل الزائد»، وهي كلمة (Polymorphism) يعود إلى اللغة اليونانية، وهي كلمة (PolymorphiC) التي تعني نماذج متعددة، وتتكون من كلمتين هما: (Poly) ومعناها تعدد، وكلمة (Morphos)، ومعناه نموذج (أو شكل). كما يُسمَّى هذا المصطلح في بعض الأحيان بمصطلح «التحميل الزائد للعوامل» (Operator Overloading). ويعني مفهوم هذا المصطلح أن مُسمَّى المعامل نفسه (أو الرمز) يمكِن أن يرتـبط بأكثر من طريقة تنفيذ للمعامل (أو الرمز)؛ وذلك حسب نوعية الأشياء التي يُطبق عليها المعامل

(أو الرمز). ومن أكثر الأمثلة شيوعاً لشرح هذا المفهوم؛ هو مثال عملية الضرب أو الجمع أو القسمة، في لغات البرمجة. فعملية الجمع، على سبيل المثال، يُرمَز لها بالرمز «+». ويمكِن تطبيق معامل الجمع على الأعداد الصحيحة (Integers)، والأعداد الحقيقية (Real) على حَدٍّ سواء، على الرغم من أن طريقة إجراء عملية الجمع في الأعداد الصحيحة تختلف عن طريقة إجراء عملية الجمع في الأعداد الحقيقية؛ من حيث الخوارزميات التي تُستخدَم لتنفيذ كلٍّ من العمليتين داخل الحاسب الآلي. ويتمُّ تحديدُ طريقة التنفيذ، أيْ: الخوارزمية المناسبة، لإجراء عملية الجمع من خلال نوعية بيانات العددين (أو المتغيرين) اللذين ستُجرَى عليهما عملية الجمع. ويعني هذا أن رمز عملية الجمع في لغات البرمجة قد يرتبط بأكثر من طريقة للتنفيذ، ومن ثم فهو محمل بشكل زائد (Overloaded). وبهذا الأسلوب نفسه يمكِن تعريف العوامل (أو الرموز) في نظم قواعد البيانات الشيئية؛ إذ إنه من الممكن أن يتمَّ الدمجُ (أو المزج) بين شيئين من نوعية بيانات «صورة» (Image)، على سبيل المثال، باستخدام رمز عملية الجمع؛ أو استخدام رمز عملية الجمع لإجراء عملية جمع بين أعداد مركبة (Complex Numbers). وفي مثل هاتين العمليتين يجب تعريف فئة صنف (Class) لكلٍّ من نوع الصور (Image)، ونوع الأعداد المركبة (Complex Number)؛ بحيث يتمُّ تعريف عملية الجمع من ضِمن العمليات التي بإمكان كلِّ صنف أن يقوم بتنفيذها. ويجب أيضاً برمجة طريقة تنفيذ عملية الجمع الخاصة بكلِّ صنف داخل فئة الصنف، بالإضافة إلى تعريف نوعية بيانات الصور والأعداد المركبة وهياكلها داخل صنف كلٍّ منها.

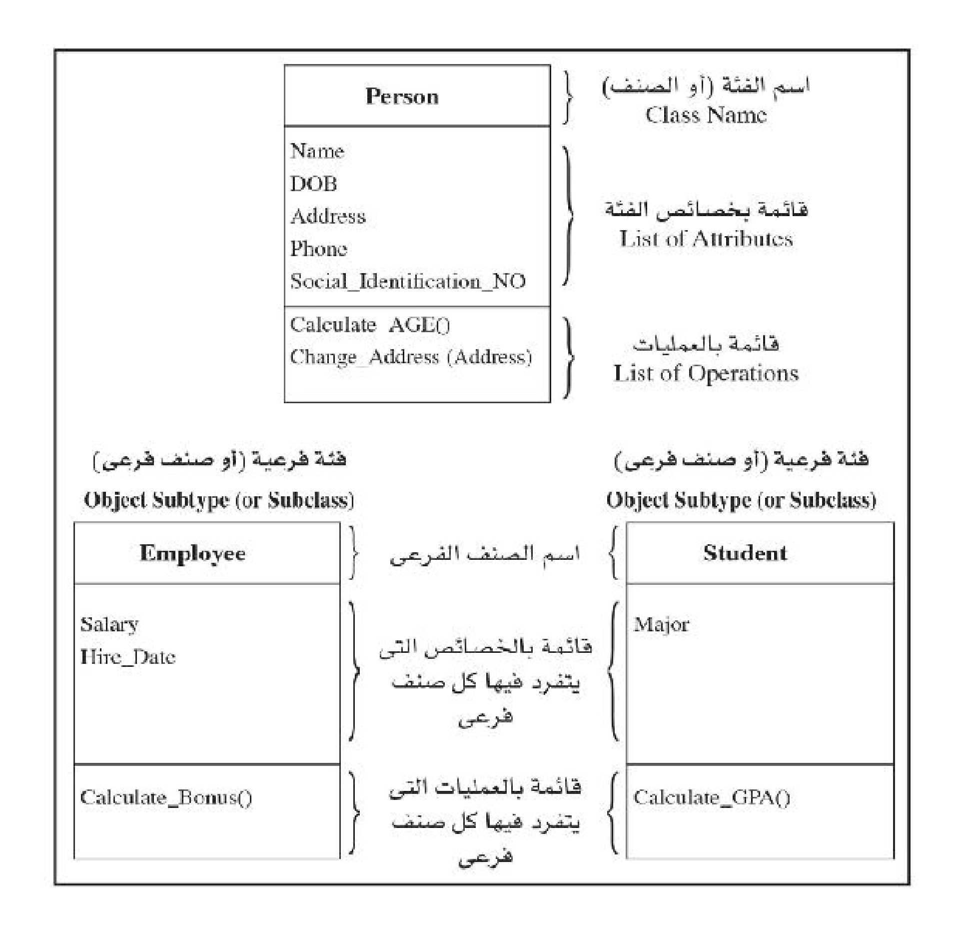
9-2-1-5 هرميات الأصناف والتوريث (Class HierarChies and InheritanCe):

توفر نظمُ قواعد البيانات الشيئية، شأنها شأن بقية أنواع نظم قواعد البيانات، القدرة على تصنيف الأشياء حسب الفئة (أو الصنف) الذي تتبعه هذه الأشياء؛ بحيث يتمُّ تجميع الأشياء المتشابهة ضِمن فئة (أو صنف) واحد، كما أسلفنا أعلاه. وتشترك الأشياء التابعة لفئةٍ (أو صنفٍ) ما في الخصائص نفسها وفي نفس أنواع العمليات التي من الممكن أن تُجرَى عليها بالإضافة إلى نوعية الارتباطات التي من الممكن أن ترتبط بها مع أنواع أخرى في قاعدة البيانات. وبالإضافة إلى ذلك؛ فإن نظم قواعد البيانات الشيئية تسمحُ بتعريف أنواع

(أو أصناف) جديدة تنحدرُ (أو تشتق) من أنواع (أو أصناف) سبق أن تمَّ تعريفها. ويُعْرَف هذا المبدأ في نظم قواعد البيانات الشيئية بـهرميات الأنواع (أو الأصناف).

وكما أسلفنا أعلاه أيضاً؛ يتم تعريف الفئة (أو الصنف) من خلال تسميتها وتعريف خصائصها والعمليات (أو الطرق) التي من الممكن أن تُجرَى (أو تُنفَّذ) عليها. ويُعَدُّ هذا النوع من الفئات (أو الأصناف) أبسط أنواع الفئات (أو الأصناف) التي يمكِن توصيفها في نظم قواعد البيانات الشيئية. ومن أمثلة الفئات (أو الأصناف) هذه؛ فئةُ الطالب (Student) التي تمَّ توصيفها في الجزء 9-2-1-2. غير أن النوع الفرعي (Subtype (or SubClass)) يُعَدُّ مفهوماً مفيداً عندما نرغب في إنشاء فئة (أو صنف) جديد مماثل لنوع (أو صنف) موجود أصلاً؛ ولكنه غير مطابق له تماماً. في هذه الحالة؛ يمكن تعريف النوع الفرعي؛ بحيث يرث الخصائص كافةً والعمليات المعرفة في النوع الموجود (أو المعرف) أصلاً، وإضافة خصائص وعمليات جديدة للنوع الفرعي غير موجودة في النوع المعرف أصلاً. وبشكلٍ عام، يرث النوع الفرعي الخصائص كافةً والعمليات المعرفة في النوع الرئيسي؛ بالإضافة إلى إمكانية تعريف خصائص وعمليات مرتبطة بـالنوع الفرعي فقط. وبهذه الطريقة يجب تعريف وبناء الخصائص والعمليات المتعلقة بالنوع الفرعي فقط دون الحاجة إلى تعريف وبناء العمليات المعرَّفة أصلاً بـالنوع الرئيسي. فعلى سبيل المثال: يمكِن تعريف «صنف الطالب» (Student Class) و«صنف الموظف» (Employee Class) على أنهما صنفان فرعيان من «صنف الشخص» (Person) على أن يكون لكلٍّ منهما الخصائص والعمليات التي يتفرد بها عن النوع الأصلي الذي أُستُمِدَّ (أو أُشتُقَّ) منه، كما هو مُوضَّح في الشكل رقم (9-4).

وبالطريقة نفسها الموضَّحة في الشكل رقم (9-4)، من الممكن أن يتمَّ تعريف أنواع فرعية جديدة من الأنواع الفرعية التي تمَّ تعريفها. وبهذه الطريقة تتكوَّن لدينا هرمية من الأنواع (أو الأصناف) الفرعية، كلٌّ منها يرث الخصائص والعمليات المعرَّفة ضمن الأنواع (أو الأصناف) التي تعلوه في الهرمية. ومن أهمِّ مَيْزات هرميات الأصناف مبدأ إعادة الاستخدام (Reusability)؛ بحيث يمكن إعادة استخدام الخصائص والعمليات التي تعرف في نوع (أو صنف) معين، في جميع الأنواع (أو الأصناف) التي تنحدر (أو تُشتَق) من النوع (أو الصنف) دون حاجة إلى تعريفها أو إعادة بنائها.



شكل رقم (9–4): مثال توضيحي للفئات (أو الأصناف) الفرعية.

3-9 قواعد البيانات «العلاقية - الشيئية» ((ObjeCt-Relational Database Systems:

يُعَدُّ النموذج العلاقي، ونظم إدارته، من أكثر النماذج شيوعاً في المنظمات الحديثة في وقتنا الراهن في بناء النظم المعلوماتية المبنية على نظم قواعد البيانات؛ وذلك على اختلاف طبيعة الأعمال التي تزاولها هذه المنظمات. وتُعزَى أسباب نجاح النموذج العلاقي وسعة انتشاره لما يلي:

1- سهولة المفاهيم الأساسية للنموذج العلاقي، واستناده إلى أسس رياضية صلبة (كما أوضحناه في الفصل الرابع).

2- يجسِّد النموذج العلاقي مفهوم عدم اعتمادية البيانات (كما تمَّ توضيحه في الفصل الأول من الكتاب).

3- يتوفَّر للنموذج العلاقي لغة تداول قوية، وذات مواصفات قياسية (وهي لغة الاستفسار البنائية التي تمَّ شرح مكوِّناتها الأساسية في الفصلين السابع والثامن).

4- يتوفَّر للنموذج العلاقي نظم إدارة قواعد بيانات ذات تقنيات ناضجة تمَّ تصميمها وتطبيقها لفترة طويلة من الزمن بما في ذلك نظم للتحكم في التزامن ونظم للتحكم في الاستعادة (أو التشافي).

وعلى الرغم من نقاط القوة السابقة للنموذج العلاقي؛ فإن هذا النموذج لا يُعَدُّ مناسباً للتعامل مع أنواع البيانات المعقدة، مثل: الصور، ولقطات الفيديو، والصوت، والبيانات الجغرافية؛ على سبيل المثال لا الحصر. لذلك تمَّ ظهور نظم قواعد البيانات الشيئية؛ بهدف التعامل مع مثل أنواع البيانات هذه. وعلى الرغم من أن نموذج البيانات الشيئي أخذ في الانتشار وأن التقنيات المصاحبة له تبدو واعدةً؛ فإنَّ هذا النموذج ما زال يعاني نقصاً في التقنيات المصاحبة له مقارنةً بتلك المصاحبة للنموذج العلاقي. ومن ضِمن هذه التقنيات: القدرة على التعامل مع البيانات بكفاءة عالية، ووجود نظم فعَّالة للاستعادة (أو التشافي)، ووجود نظم مَرِنة للتحكُّم في التزامن.

ويُمكِن أن نستخلص أن لكلٍّ من النموذجين نقاط قوة تميُّزه عن النموذج الآخر، وأن نقاط قوة أحد هذين النموذجين تُعَدُّ نقاط ضعف في النموذج الآخر. وقد حدا هذا بالشركات المطوِّرة لنظم إدارة قواعد البيانات إلى تبنِّي نظم مُهجَّنة (Hybrid Systems)؛ تهدف إلى الحصول على أفضل الميزات التي يوفرها كلٌّ من النموذجين. وأصبحت هذه النظم تُسمَّى نظم قواعد البيانات «العلاقية - الشيئية». كما أن غالبية الشركات المطوِّرة لنظم إدارة قواعد البيانات العلاقية تتبنَّى ضِمن النسخ الحديثة لمنتجاتها مفاهيمَ متوافقةً مع مفاهيم نظم قواعد الشيئية، مثل: الشيء (ObjeCt)، والتغليف (EnCapsulation)، والتحميل الزائد (Polymorphism)، والتوريث (InheritanCe). ومن ضِمن أكبر هذه الشركات التي تبنَّت هذا التوجُّه شركة أي بي إم (IBM) في منتجها المعروف باسم دي بي تو (DB2)، وشركة إنفورميكس (Informix) في منتجها المعروف باسم داينميك سيرفر (DynamiC Server)، وشركة أوراكل (OraCle) في منتجها أوراكل 8 (OraCle 8) والنسخ اللاحقة لهذه النسخة من المنتج.

9–3–1 مفاهيم قواعد البيانات «العلاقية – الشيئية»:

يُعَدُّ نظام إدارة قاعدة البيانات «العلاقية - الشيئية» نظاماً يدعمُ كلاً من خصائص النموذج العلاقي وخصائص النموذج الشيئي بشكلٍ متكاملٍ (Frank, 1995). وبذلك؛ فإنه يمكن تعريف البيانات وفق النموذج العلاقي ووفق النموذج الشيئي والتعامل مع هذه البيانات من خلال واجهة مشتركة (مثل: لغة الاستفسار البنائية (SQL)). أمَّا البنية التحتية لهذه النظم؛ فهي وفق النموذج العلاقي. ويعني هذا أن قاعدة البيانات تبدو لمبرمجي نظم التطبيقات (وبقية المستفيدين من النظام) على أنها «علاقية - شيئية»؛ ولكنها في الواقع تخزَّن ويتمُّ التعامل معها داخلياً على أنها علاقية فقط. لذلك؛ فإن هناك بعض التكلفة الإضافية في هذه النظم نتيجةً لضرورة المطابقة (أو التحويل) (Mapping) بين العلاقات (Relations) والأشياء (ObjeCts). ويُعزَى السبب وراء هذه التكلفة الإضافية إلى أن هذه النظم قد تمَّ بناؤها أساساً باعتبارها نظماً لإدارة قواعد البيانات العلاقية ولم تكن مُصمَّمة لدعم أيٍّ من الخصائص التي يوفرها النموذج الشيئي؛ ولكنه تمَّ تطويرها لاحقاً لتدعم بعض خصائص النموذج الشيئي. لذلك؛ فإننا نرى أن مثل هذه النظم تُسمَّى أحياناً بالنظم العلاقية المطورة (Extended Relational)؛ وذلك للدلالة على هذا الواقع.

9-3-1-1 خصائص قواعد البيانات «العلاقية – الشيئية»:

توفر لغة الاستفسار البنائية ثلاثة أصناف من البيانات، وهي: (1) الرَّقمية مثل الأعداد العشرية (DeCimal)، والأعداد الصحيحة (Integer)، (2) السلاسل الحرفية (CharaCter)، (3) الوقتية (Temporal)، مثل: التاريخ (Date)، والوقت (Time). غير أن الكثير من نظم التطبيقات الحديثة تتطلب تخزين وتداول أنواع أخرى من البيانات التي لا يمكن التعامل معها بسهولة من قِبل نظم إدارة قواعد البيانات العلاقية، مثل: الوثائق، والصور، وبصمات الأصابع، والخرائط؛ على سبيل المثال لا الحصر. وللتعامل مع مثل أنواع البيانات هذه؛ فإنه يتطلب من نظام إدارة قاعدة البيانات توفير إمكانية تعريف أنواع بيانات جديدة، حسب احتياجات تطبيقات المنظمة؛ بالإضافة إلى تلك المتوفرة فيها بشكلٍ أساسي. كما أن توفير إمكانية تعريف أنواع بيانات جديدة من قِبل نظام إدارة قاعدة البيانات يتطلب بدوره من النظام توفير إمكانية تعريف عمليات جديدة يمكِن إجراؤها على أنواع البيانات الجديدة، شبيهةً بتلك العمليات التي يوفرها النظام على أنواع البيانات الأساسية، مثل: الجمع والطرح بالنسبة للأعداد والبيانات الوقتية، وعمليات

المقارنة بالنسبة للسلاسل الحرفية. فعلى سبيل المثال: قد يستلزمُ تعريف نوع جديد من البيانات وليكن صورة (Image) (أو خريطة (Map)) توفير عملية جديدة يمكن تنفيذها على هذا النوع من البيانات، مثل: عملية التكبير (Zoom In)، وعملية التصغير (Zoom Out).

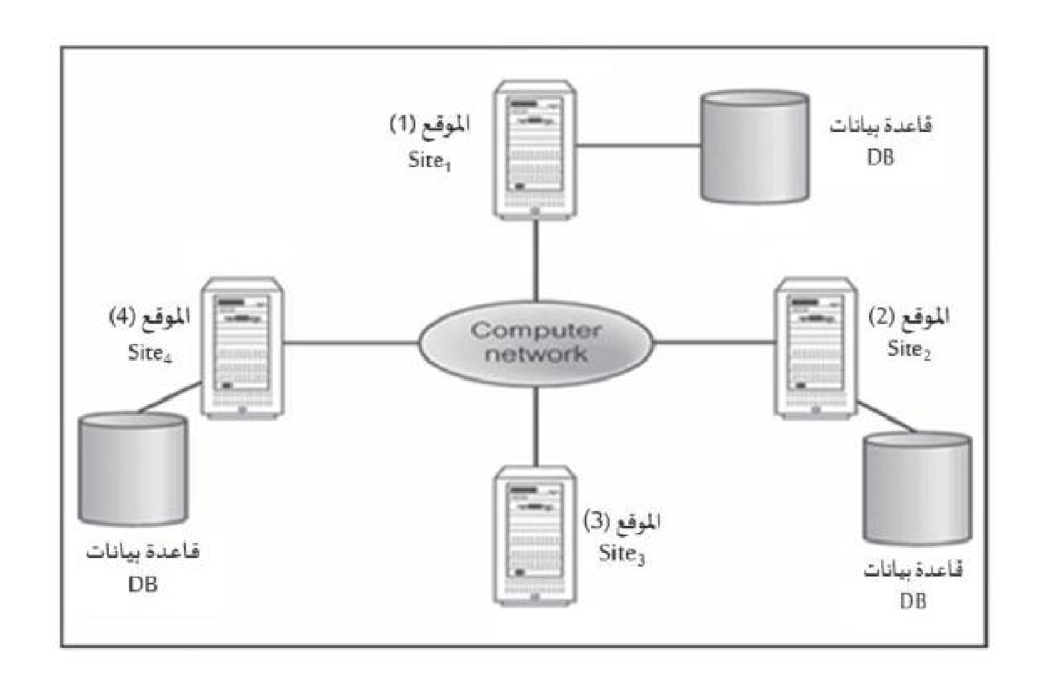
كما أن إمكانية تعريف أنواع بيانات جديدة يتطلَّب من نظم إدارة قواعد البيانات توفير لغة فعَّالة للتعامل مع البيانات شبيهةً بلغة الاستفسار البنائية؛ بحيث تُمكّن من تعريف، وتداول البيانات. ولأن الهدف من قواعد البيانات «العلاقية - الشيئية» هو دمج أفضل خصائص النموذج العلاقي وأفضل خصائص النموذج الشيئي ضِمن نظام إدارة قاعدة البيانات؛ فإن أهمَّ خصائص هذا الهجين الناتج من دمج النموذجين، ما يلي:

1- نسخة مُطوَّرة من لغة الاستفسار البنائية تمكِّن من تعريف وتداول البيانات؛ سواء كانت مُخزَّنة في جداول علاقية أو فئات أصناف (أو أشياء) (ObjeCt Types).

2- توفير الدعم لخصائص النموذج الشيئي، مثل: التوريث، والتحميل الزائد، وتعريف أنواع بيانات جديدة (User Defined Data Types)، وطرق الوصول إلى الأشياء (Navigational ACCess).

4-9 قواعد البيانات الموزَّعة (Distributed Database Systems):

قاعدة البيانات الموزَّعة عبارة عن قاعدة بيانات منطقية واحدة موزَّعة مادياً على مجموعة من الحاسبات الآلية في مواقع مختلفة ترتبط فيما بينها بشبكة اتصالات (Stallings, 1993). وتُعَدُّ قواعد البيانات الموزَّعة مفيدةً عندما يكون للمنظمة مجموعةٌ من المواقع (أو الفروع) الموزَّعة في مناطق مختلفة؛ بحيث يتم توزيع البيانات بشكل أقرب ما يكون من المستفيدين الذين يتعاملون معها. ويمثل الشكل رقم (9-5) نموذجاً مبسطاً لقاعدة بيانات موزَّعة تتكون من أربعة مواقع (Sites) تحتوي ثلاثةٌ منها على البيانات المكوِّنة لقاعدة البيانات، في حين لا يحتوي الموقع رقم 3 (Site 3) على أيِّ بيانات تتعلق بمحتويات قاعدة البيانات، وإنما يُستَخدَم الموقعُ مصدراً للتعليمات التي تتفاعل مع قاعدة البيانات الموزَّعة على المواقع الثلاثة الأخرى. أمَّا المواقع الثلاثة الأخرى؛ فإنها تستخدم مصادر للتعليمات التي تتفاعل مع قاعدة البيانات بالإضافة إلى احتواء كلٍّ منها على جزءٍ من قاعدة البيانات (الموزَّعة). وترتبط المواقع الأربعة فيما بينها بشبكة اتصالات لنقل البيانات.



شكل رقم (9–5): نموذج مبسط لقاعدة بيانات موزَّعة.

وتتمُّ إدارة قاعدة البيانات (الموزَّعة) بشكلٍ مركزي باعتبارها مورداً من موارد المنظمة. وفي الوقت نفسه؛ تتمُّ الاستفادة من المرونة التي تقدمها نظم قواعد البيانات الموزَّعة في تنفيذ العمليات التي تُجرَى على قاعدة البيانات. ويُوجَد العديد من الحالات التي تشجع على استخدام نظم إدارة قواعد البيانات الموزَّعة التي يأتي من ضِمنها ما يلي (Hoffer et al, 2015):

- التوزيع الجغرافي: من الطبيعي أن تنتشر الأقسام والإدارات في المنظمات الحديثة في مواقع متباعدة جغرافياً عن بعضها، وفي بلدان مختلفة أحياناً. وفي هذه الحالة تتولَّد رغبة عند هذه الأقسام والإدارات في فرض صلاحياتها على نظمها المعلوماتية التي تستدعي توفر هذه النظم المعلوماتية بمقربة من الأقسام والإدارات التي تتبعها هذه النظم حتى تتحكم فيها بشكلٍ كامل.

- المشاركة: تُوجَد هناك حاجةٌ إلى المشاركة في البيانات عندما تتوزَّع النظم المعلوماتية والبيانات التابعة لهذه النظم في أماكن مختلفة؛ حتى يتسَنَّى اتخاذ القرارات التي تتعلق بأكثر من قسم وإدارة من الوحدات الإدارية التابعة للمنظمة. وتتجَلَّى مثل هذه الحالة عند اندماج المنظمات (أو الشركات) بعضها مع بعض.

- التكلفة: إن وجود البيانات ونظم التطبيقات بالقرب من الأقسام والإدارات التي تحتاج إليها بشكلٍ مكثف يكون في الغالب أقل تكلفةً من الناحية الاقتصادية؛ إذ إن إرسال كميات كبيرة من الأوامر الحاسوبية من مكان لآخر (عبر وسائل الاتصالات) يُعَدُّ أمراً باهظ التكلفة. بالإضافة إلى ذلك؛ فإن اعتماد نظم التطبيقات على نظم الاتصالات قد يكون مصدراً من مصادر المخاطرة؛ نظراً لإمكانية تعطل نظم الاتصالات ولفترات طويلة. لذا فإن توزيع البيانات؛ بحيث تكون على مقربةٍ من الوحدات الإدارية (والمستفيدين) الذين يحتاجون إليها بشكلٍ مكثفٍ يكون أكثر موثوقية مقارنةً بتركيزها في مكانٍ واحد حتى لو تعطل جزءٌ من نظام الاتصالات أو بعض الأجهزة التي تحتوي على بيانات المنظمة.

- عدم التجانس: يتوفر لدى المنظمات الحديثة أنواعٌ مختلفةٌ من نظم التطبيقات التي قد تعتمد على أنواع مختلفة من نظم إدارة قواعد البيانات؛ بحيث إن كلَّ نوع منها قد يكون هو أفضل ما يكون للنظام الذي تم اقتناؤه من أجله. وفي هذه الحالة يمكِن تعريف نظام قواعد بيانات موزَّع يعمل على ربط التطبيقات المختلفة مع بعضها.

- تكرارية البيانات: تُعَدُّ تكرارية البيانات (Data RepliCation) إحدى الإستراتيجيات المتبعة لاستعادة البيانات التي تتعرَّض للتلف، ولتمكين المستفيدين من مزاولة تعاطيهم مع البيانات حتى لو تعطل المصدر الرئيسي الذي تمَّ تخزين البيانات عليه. وتُعَدُّ تكرارية البيانات على أكثر من جهاز حاسب آلي أحد أشكال نظم قواعد البيانات الموزَّعة.

ويُعَدُّ من الأهداف الرئيسية لنظم قواعد البيانات الموزَّعة؛ توفيرُ خدمة وصول المستفيدين للبيانات المخزَّنة في مواقع مختلفة. وللوصول إلى الهدف؛ يجب في نظام إدارة قواعد البيانات الموزَّعة توفير ما يُعرَف بالشفافية المكانية (LoCation TransparenCy) التي تَعنِي أنه ليس من الضروري أن يكون المستخدِم على دراية بالمواقع التي تحتوي على البيانات التي يتعاطى معها من خلال التعليمات التي يصدرها للنظام؛ بل إن كلَّ تعليمة يصدرها المستخدِم سيتمُّ إرسالها من قبل نظام إدارة قاعدة البيانات بشكل تلقائي (دون علم من المستخدِم) للمواقع ذات الصلة بتنفيذ التعليمة. وفي الحالة المثلى لنظم قواعد البيانات الموزَّعة يكون المستخدِم غير مدرك لطريقة توزيع البيانات ومواقعها، وأن كل البيانات على اختلاف مواقعها تشكِّل قاعدة بيانات منطقية واحدة. وفي مثل

هذه الحالة المثلى؛ فإن التعليمة الواحدة قد تقوم بتجميع بيانات موجودة مادياً في مواقع مختلفة للحصول على النتيجة النهائية للتعليمة.

ومن الأهداف الرئيسية الأخرى لنظم قواعد البيانات الموزَّعة؛ الاستقلال الذاتي (LoCal Autonomy) للمواقع المختلفة لقاعدة البيانات التي يُقصَد بها القدرة على إدارة كل موقع من مواقع قاعدة البيانات بشكلٍ مستقل؛ بحيث يستطيع كلُّ موقع العمل باستقلالية عن بقية المواقع؛ وذلك عند حدوث عطل لشبكة الاتصالات أو للمواقع الأخرى. وعند وجود الاستقلال الذاتي؛ فإنه بإمكان كلِّ موقع أن يتحكم في البيانات المخزَّنة فيه، وأن يدير أمنها، وأن يقوم بتسجيل المعاملات التي ترد إليه وتنفيذها، وأن يقوم باستعادة بـياناته عند حدوث عطل فيه دون وجود موقع مركزي يُشرِف على أعماله هذه. وبذلك؛ فإن البيانات تُعَدُّ مملوكةً محلياً ضِمن الموقع ومُدارة من قِبله، على الرغم من أن البيانات المخزَّنة فيه يمكِن التعاطي معها من قِبل مواقع أخرى.

ويُوجَد لقواعد البيانات الموزَّعة العديدُ من المَيْزات مقارنةً بنظم قواعد البـيانات المركزية. ومن بين أهمِّ هذه المَيْزات، ما يلي:

- الموثوقية (Reliability) والتواجد (Availability): تُعرَّف الموثوقية، بشكلٍ عام، بأنها احتمالية وجود النظام بشكلٍ عاملٍ في لحظة زمنية ما؛ في حين يُعرَّف التواجد بأنه احتمالية وجود النظام بشكلٍ عامل وبـشكلٍ متواصل ضِمن مدى زمني ما. وعلى النقيض من نظم قواعد الـبيانات المركزية التي يتأثر فيها جميعُ المستفيدين والتطبيقات عند تعطل النظام؛ بحيث يصبح من المتعذّر الوصول إلى كلِّ البيانات المخزَّنة في قاعدة البيانات؛ فإن حدوث عطل لموقع (أو أكثر) في نظم قواعد البيانات الموزَّعة أو لجزءٍ من شبكة الاتصالات
لن يؤثر في جميع المستفيدين والتطبيقات؛ وذلك لأنه من الممكن أن يتمَّ التعاطي مع الجزء الذي ما زال عاملاً من قاعدة البيانات، ولكن بشكلٍ وظيفي أقل مما كانت عليه قبـل حدوث العطل. ويعني هذا أن نظم قواعد الـبيانات الموزَّعة تقوم بتحسين كلٍّ من موثوقية وتواجد النظام.

- التحكُّم الذاتي (LoCal Control): يُشجِّع توزيع البيانات المجموعات المختلفة من المستفيدين على ممارسة تحكمهم في البيانات الخاصة بهم بشكلٍ أكبر؛ مما يعني تحسين تكامل البـيانات وإدارتها. كما يمكن استخدام العتاد المادي المناسـب من الحاسبات الآلية وملحقاتها مع البيانات الخاصة بكلِّ مجموعة من المستفيدين، وبما يتناسب مع احتياجاتهم من القدرات

الحاسوبية. وبهذه الطريقة يمكِن لكلِّ مجموعة التعامل مع البيانات الخاصة بها وفق احتياجاتها الحاسوبية التي تتناسب معها، وفي الوقت نفسه، التعاطي مع البيانات التابعة للمجموعات الأخرى من المستفيدين عند الحاجة إلى ذلك.

- تخفيض تكلفة الاتصالات: تُمكِّن نظم قواعد البيانات الموزَّعة من تخزين البيانات بشكلٍ أقرب ما يكون من المستفيدين الذين يحتاجون إلى التعامل معها بشكلٍ مكثف. ويعني هذا تخفيض تكلفة استخدام وسائل الاتصالات مقارنةً بنظم قواعد البيانات المركزية.

- تحسين أداء النظام: يُمكِن توزيع البيانات بشكلٍ يُلبي احتياجات المستفيدين في كل موقع. وعند توزيع البيانات بهذا الشكل؛ يصبح بالإمكان تنفيذ تعليمات المستفيدين في مواقعهم دون الحاجة إلى استخدام وسائل الاتصالات أو تركيز تنفيذ التعليمات في حاسب آلي واحد؛ مما يحسِّن أداء النظام بشكلٍ كبير. كذلك يُمكِن تجزئة التعليمات المعقدة وتنفيذ التعليمات الجزئية الناتجة على أكثر من حاسب آلي بشكلٍ متزامن، أيْ: في الوقت نفسه، على أكثر من حاسب آلي؛ مما يُسهِم في تحسين أداء النظام بشكلٍ أكبر.

- سهولة التوسُّع في النظام: تتوافق نظم قواعد البيانات الموزَّعة مع حاجة المنظمات الحديثة للتوسع المستمر في طاقاتها التخزينية للبيانات؛ وذلك عند زيادة أحجام البيانات، أو في زيادة أعداد تطبيقاتها الحاسوبية؛ إذ إنه بالإمكان زيادة عدد الحاسبات الآلية (لزيادة طاقتها التنفيذية) وطاقاتها التخزينية كلما استدعت الحاجة إلى التوسُّع. وهذه الطريقة تُعَدُّ أكثر اقتصاداً من التوسُّع في نظم قواعد البيانات المركزية التي تستدعي تغيير الحاسب الآلي (وبعض ملحقاته أحياناً). بالإضافة إلى ذلك؛ فإن التوسُّع في نظم قواعد البيانات الموزعة لا يؤدي إلى انقطاع الخدمة عن المستفيدين أو تأثر عملهم مقارنةً بنظم قواعد البيانات المركزية التي قد تستدعي مثل هذا الانقطاع أو التأثر.

9–4–1 خيارات توزيع البيانات:

عند استخدام نظم قواعد البيانات الموزَّعة يجب تحديد الأماكن (أو المواقع) التي سيتمُّ فيها تخزين كلِّ مجموعة من البيانات المكوِّنة لقاعدة البيانات. ولأن الجدول أصغر وحدة منطقية يمكِن الحديث عنها في نظم قواعد البيانات؛ فإن الجدول يُعَدُّ أصغر وحدة منطقية يمكِن التحدُّث عنها عند توزيع البيانات على مواقع نظم قواعد البيانات الموزعة. كما يمكِن تقسيم كلِّ جدول إلى مجموعة

من الجداول الجزئية؛ بحيث يحتوي كلُّ جدول جزئي على مجموعة من السجلات التي تتحلى بخصائص مشتركة فيما بينها. وفي هذه الحالة يُعَدُّ كلُّ جدول جزئي جدولاً مستقلاً بذاته عن بقية الجداول الجزئية المشتقة من الجدول الرئيسي نفسه. ويمكِن توزيع جداول قاعدة البيانات في نظم قواعد البيانات الموزَّعة على المواقع المكونة للنظام وفق أربع إستراتيجيات، هي:

- تكرار البيانات (Data RepliCation).
- التقسيم الأفقي (Horizontal Partitioning).
- التقسيم الرأسي (VertiCal Partitioning).
- الجمع بين الخيارات السابقة.

9-4-1-1 تكرار البيانات:

يُعَدُّ تكرارُ البيانات واحداً من الخيارات التي يزداد انتشارها يوماً بعد آخر. ويُقصَد بتكرار البيانات تخزينُ نسخ كاملة من قاعدة البيانات في أكثر من موقع. ويوفر هذا الخيار فرصةً للانتقال من قواعد البيانات المركزية إلى قواعد البيانات الموزَّعة ذات التكلفة الأقل؛ بحيث تكون البيانات قريبةً مكانياً من المستفيدين منها. وتحسِّن هذه الإستراتيجية من إستراتيجيات توزيع البيانات من الاعتمادية والتواجد؛ إذ إن النظام سيستمرُّ في العمل ما دام قد وُجِدَ موقعٌ واحدٌ (على الأقل) قيد العمل. كما يُحسِّن هذا الخيار أداء النظام؛ لأن نتائج تعليمات الاستفسار التي تصدر للنظام تكون قريبةً من مكان صدور التعليمات نفسها. غير أن خيار تكرارية البيانات يعاني مشكلةً مع تعليمات التعديل؛ إذ إن التعليمات يجب أن تُنفَّذ في جميع المواقع التي تحتوي على نسخ من قاعدة البيانات؛ وذلك حتى تكون البيانات المخزَّنة في المواقع المختلفة متوافقاً بعضها مع بعض. ولهذا السبب؛ فإن تكرار البيانات يُستخدَم عادةً في نظم قواعد البيانات الموزَّعة التي يغلب فيها عمليات الاستفسار عوضاً عن تعليمات التعديل.

وبينما يمثل التكرارُ الكامل للبيانات إحدى النهايتين القصويين لإستراتيجية تكرار البيانات؛ فإن عدم التكرار لأيٍّ من البيانات يمثل الجانب الآخر من النهايتين. وبين هاتين النهايتين القصويين تُوجَد مساحةٌ كبيرة للتكرار الجزئي للبيانات (Partial Data RepliCation) الذي يتمُّ من خلاله

تكرارُ بعض أجزاء قاعدة البيانات، في حين لا يتم تكرار أجزاء أخرى. ويتم توزيع الأجزاء الرئيسية لقاعدة البيانات أو المُستنسَخة منها على مواقع مُحدَّدة في النظام. ويتمُّ اختيار المواقع ودرجة التكرار بناءً على الاعتمادية والتواجد المستهدفتين؛ بالإضافة إلى نوعية وتكرارية العمليات المتوقع تنفيذها على النظام. وتُعَدُّ عملية إيجاد التوزيع الأمثل للبيانات على المواقع المختلفة للنظام؛ من العمليات المعقدة نسبياً؛ وذلك بسبب وجود العديد من العوامل التي يجب أخذها بعين الاعتبار عند إجراء عملية التوزيع.

9-4-1-2 التقسيم الأفقي:

يُقصَد بالتقسيم الأفقي تجزئة الجدول إلى مجموعات من السجلات؛ بحيث تتحَلَّى كل مجموعة من السجلات بخاصية (أو قيمة) مُحدَّدة لحقل أو أكثر من حقول الجدول. وغالباً ما يتمُّ التقسيم الأفقي للجداول وفق قيمةٍ واحدة من حقول الجدول. فعلى سبيل المثال: من الممكن أن يتمَّ تقسيمُ جدول أعضاء هيئة التدريس في الجامعة الأهلية أفقياً وفق قيم حقل «رمز القسم». وبهذه الطريقة يتمُّ تقسيم الجدول إلى مجموعات من السجلات بحيث تتحلَّى كلُّ مجموعة بقيمة مُحدّدة لقيمة حقل «رمز القسم». وبعد تقسيم الجدول إلى مجموعات يُمكِن توزيع الأجزاء الناتجة؛ بحيث تكون على أجهزة الحاسب الآلي التي تكون أقرب ما يكون من القسم العلمي الذي تتبعه كلُّ مجموعة. وبهذه الطريقة يمكِن أن تتعامل كلُّ مجموعة من المستفيدين التابعين لقسم علمي ما مع بيانات أعضاء هيئة التدريس التابعين للقسم نفسه بشكلٍ أسرع مما لو كانت البيانات مركزة في موقع واحد.

9-4-1-3 التقسيم الرأسي:

يُستخدَم التقسيمُ الرأسي لتوزيع الحقول التي يحتويها جدولٌ ما على أكثر من جدول مع تكرار المفتاح الرئيسي للجدول الأساسي في جميع الجداول الناتجة عن عملية التقسيم. ومن أمثلة التقسيم الرأسي، وكما أسلفنا في الجزء المتعلق بالتصميم المادي، تقسيم جدول الموظفين إلى جدولين، هما: جدول يحتوي على المفتاح الرئيسي لجدول الموظفين، بالإضافة إلى الحقول التي تحتوي على البيانات المتعلقة بالجوانب الإدارية للموظفين، في حين أن الجدول الثاني يحتوي على المفتاح الرئيسي لجدول الموظفين بالإضافة إلى الحقول التي تحتوي على البيانات المتعلقة بالجوانب المالية للموظفين. وبعد عملية التقسيم تتمُّ عملية توزيع الجداول الناتجة على المواقع المناسبة في

النظام. وبهذه الطريقة يمكِن الاستجابة للتعليمات المتعلقة بكلِّ نوعٍ من البيانات بشكلٍ أسرع من الرد عليها عند تخزين جميع البيانات المتعلقة بـالموظفين ضِمن الجدول نفسه. ولأنه سيتمُّ وضع بيانات كلِّ إدارة على الجهاز التابع (أو الأجهزة التابعة) للإدارة؛ فإن هذا التوزيع سيوفر لكلِّ إدارة استقلالاً ذاتياً يمكِّنها من التحكم والسيطرة على البيانات التابعة لها؛ مما يحسِّن من تناسق البـيانات وتكاملها. بالإضافة إلى أن مثل هذا التوزيع سيزيد من المحافظة على أمن البـيانات؛ إذ يُمكِّن من مَنح الصلاحيات للمستفيدين المختلفين على كلِّ جدول من الجداول الناتجة من عملية التقسيم عوضاً عن منح الصلاحيات لهم على الجدول الأصلي كاملاً.

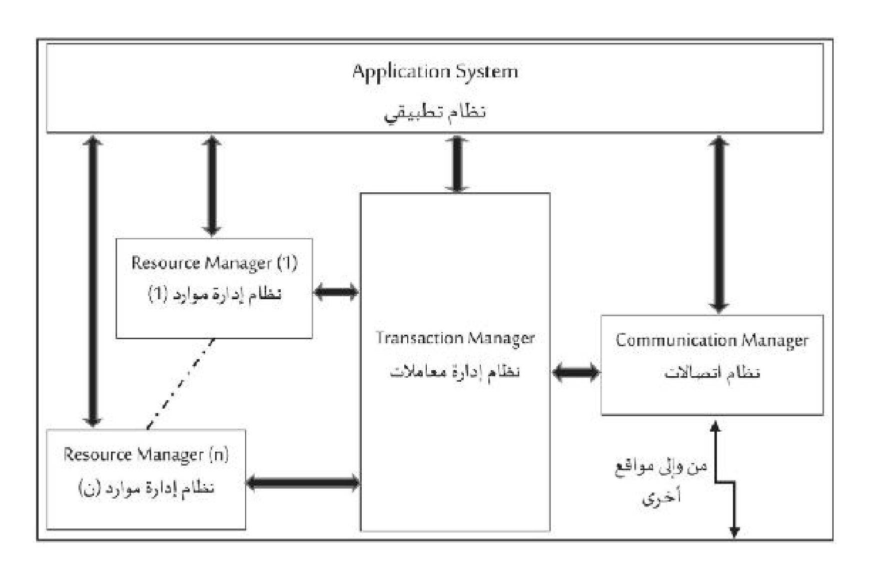
9-4-1-4 الجمع بين خيارات التوزيع:

النوع الرابع للتقسيم يتضمَّن كلاً من تكرار البيانات والتقسيم الأفقي والتقسيم الرأسي، أو أيِّ توليفات أخرى للخيارات السابقة من توزيع البـيانات. فعلى سبيل المثال: من الممكن تقسيم جدول الموظفين التابع لمنظمةٍ ما أفقياً حسب الفروع التي يتبع لها الموظفون، وبحيث يتم توزيع الجداول الناتجة من عملية التقسيم على الحاسبات الآلية التابعة للفروع المختلفة كلٌّ حسب الموظفين التابعين له. وبعد ذلك يمكن تقسيم كل جدول، في كل فرع، رأسياً حسب بـيانات الموظفين الإدارية والمالية، على افتراض وجود إدارة للشئون الإدارية وإدارة للشئون المالية في كلِّ فرع. وبهذه الطريقة يمكِن الاستفادة من المَيْزات التي توفرها قواعد البيانات الموزعة التي يأتي من أهمِّها سرعة الوصول إلى البيانات والتعامل معها من قِبل كلِّ إدارة، هذا بالإضافة إلى الاستفادة من الميزات الأخرى مثل المحافظة على الاستقلال الذاتي وأمن البـيانات. وتجدر الإشارة إلى أن الجمع بين خيارات توزيع البيانات، وخاصةً التقسيم الرأسي والتقسيم الأفقي للبيانات، لا يُعَدُّ مقصوراً على قواعد البيانات الموزَّعة؛ ولكنه يُستخدَم أيضاً في قواعد البيانات المركزية أيضاً. ويُستخدَم مثل هذا الجمع بين الخيارات، في قواعد البيانات المركزية، من قِبل إداريي قواعد البيانات في الكثير من الأحيان؛ لتحسين أداء نظم إدارة قواعد البيانات (Agarwal et al, 2004).

9-4-2 تنفيذ المعاملات الموزَّعة (ExeCution ’Distributed TransaCtions):

تُنفَّذ المعاملة في بيئة نظم قواعد البيانات الموزَّعة؛ شأنها في ذلك شأن المعاملة في بيئة قواعد البيانات المركزية، كوحدة عمل منطقية واحدة تنتهي بتثبـيت كافة نتائجها أو التراجع التام عنها، وكأنها لم تكن. ولكن المعاملة في بيئة قواعد البيانات الموزَّعة، وعلى خلاف بيئة قواعد

البيانات المركزية، تتعاطى مع بيانات موزَّعة على أكثر من موقع للبيانات. وعادةً ما تتمُّ نمذجة مكوِّنات كلِّ موقع من مواقع قاعدة البيانات الموزَّعة على أنه يتكوَّن من ثلاثة مكونات رئيسية، وهي: نظام إدارة معاملات، ونظام اتصالات، ونظام (أو أكثر من نظم) إدارة موارد بيانات (Al-Houmaily, 2014). ويوضِّح الشكل (6-9) هذه المكونات الثلاثة.



شكل رقم (9–6): المكوِّنات الرئيسية لنموذج قاعدة البيانات الموزَّعة.

1. نظام إدارة المعاملات (TransaCtion Manager (TM)): يُوجَد في كلِّ موقع من مواقع قاعدة البيانات نظامٌ واحدٌ لإدارة المعاملات. ومن مسئوليات هذا النظام تخصيص معرف (Identifier) فريد لكلِّ معاملة تبدأ في التنفيذ في الموقع. ويُعَد المعرفُ الذي يُخصَّص لمعاملةٍ ما معرفاً فريداً لا يتكرَّر ضِمن المعرفات الأخرى للمعاملات بغض النظر عن مواقع بدايات تنفيذ هذه المعاملات. وبالإضافة لذلك؛ فإن هذا النظام مسئولٌ عن مراقبة تقدُّم تنفيذ كلِّ معاملة تبدأ في موقعه، والإشراف على انتهائها، وتنسيق تشافيها عند الإخفاق في تنفيذها بشكلٍ كامل.

2. نظام الاتصالات (CommuniCation Manager (CM)): يُمكِّن هذا النظام من تناقل البيانات وتبادلها مع المواقع التي يرتبط بها من خلال تضمُّنه على بروتوكول للتواصل، وتوفير منظور داخلي، ضِمن الموقع، لقاعدة البيانات في المواقع الأخرى. كما يسمحُ هذا النظام للمعاملات التي تُنفَّذ داخلياً ضِمن الموقع من أن توسع من تنفيذ عملياتها لتشمل مواقع أخرى، ضمن بيئة قاعدة البيانات الموزَّعة، حسب احتياجاتها من البيانات. ويسمح هذا النظام كذلك لنظام إدارة

المعاملات بالموقع من القيام بمهامِّه الإدارية والتنسيقية بالتعاون مع نظم إدارة المعاملات في المواقع الأخرى.

3. نظام إدارة موارد البيانات (ResourCe Manager (RM)): يُعَدُّ هذا النظام هو النظام المسئول عن تنفيذ العمليات الفعلية على البيانات. ومن الممكن أن يكون هذا النظام أيَّ نظامٍ يسمح بمشاركة البيانات وصحتها حتى في حالة حدوث أعطال في الموقع. وعادةً ما يُفترض أن هذا النظام هو نظامٌ لإدارة قاعدة بيانات (مثل: أوراكل (OraCle)، أو دي بي تو (DB2))، رغم أنه من الممكن أن يكون أيُّ نظامٍ آخر يتحَلَّى بصفات من شأنها أن تحافظ على خصائص المعاملات. كما يمكن أن يحتوي الموقع الواحد على أكثر من نظام لإدارة الموارد.

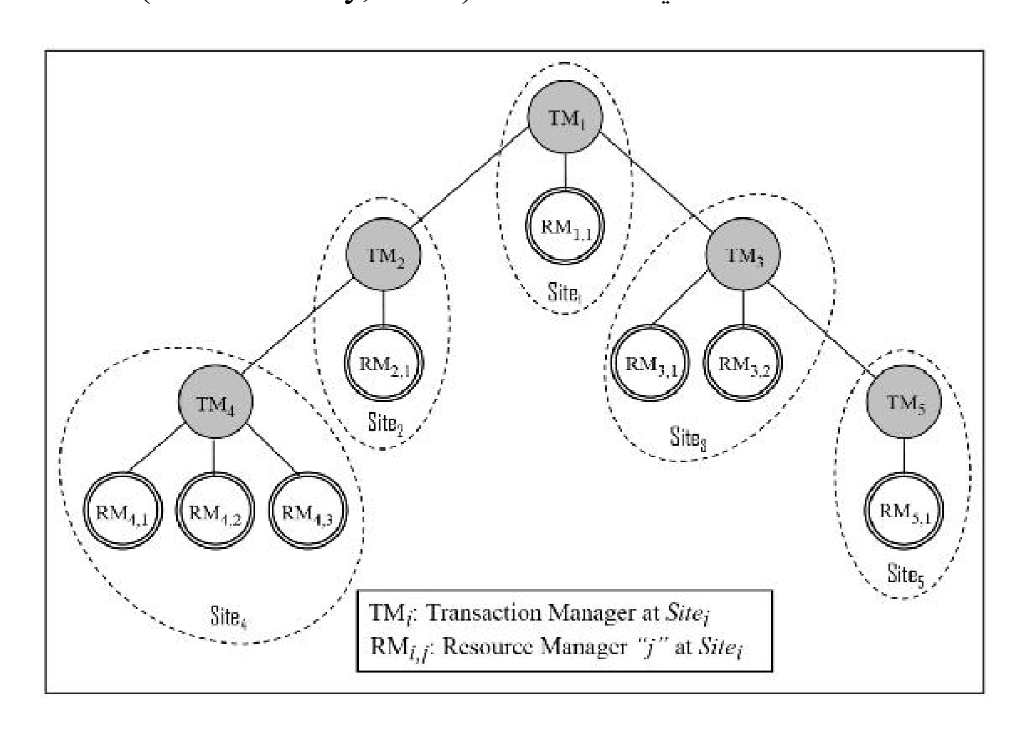
9–4–2–1 نموذج تنفيذ المعاملة (TransaCtion ExeCution Model):

عند بداية تنفيذ أيِّ معاملة صادرة من نظام تطبيقي ما (AppliCation System) في أحد المواقع؛ فإنه يتعيَّن عليها التسجيل في نظام إدارة المعاملات في الموقع الذي بدأت تنفيذها فيه قبل أن تتمكَّن من تنفيذ أيٍّ من عملياتها على قاعدة البيانات. وعندما يقوم نظام إدارة المعاملات بتسجيل المعاملة؛ فإنه يقوم بتخصيص معرف فريد لها لا يتكرر ضِمن معرفات المعاملات الأخرى التي سبق تسجيلها في كافة مواقع النظام. كما يقوم نظام إدارة المعاملات بإشعار كلِّ نظام لإدارة موارد البيانات، ضِمن موقعه، عن بداية تنفيذ المعاملة. وبهذه الطريقة يكون نظام إدارة المعاملات وجميع نظم إدارة الموارد في الموقع على علم ببداية تنفيذ المعاملة.

بعد أن تتمَّ عملية تسجيل المعاملة؛ يصبح بإمكانها أن تقوم بتنفيذ أيٍّ من عملياتها على نظم إدارة الموارد المتاحة في الموقع. وعندما ترسل المعاملة عمليةً للتنفيذ لأحد نظم إدارة الموارد؛ فإنها تتوقف عن التنفيذ؛ حتى تستلم نتيجة تنفيذ العملية. وبعدئذٍ فقط يمكنها إرسال عملية أخرى للتنفيذ. وتُسمَّى طريقة التنفيذ هذه «الطلب والاستجابة» (Request/Response Paradigm)؛ حيث إنه لا يمكن للمعاملة من تنفيذ عمليةٍ ما إلا بعد انتهاء العملية السابقة لها في التنفيذ.

وبنفس الأسلوب يُمكِن للمعاملة تنفيذ عمليات على موارد للبيانات خارج موقعها الذي بدأت تنفيذها فيه. ولكنه في هذه الحالة؛ يتوجَّب أن يكون نظام إدارة المعاملات في الموقع الذي نشأت فيه المعاملة على علمٍ، وبشكلٍ مُسبق، عن كلِّ عمل يتمُّ تنفيذه على مواقع خارجية. وعند علم نظام

إدارة المعاملات عن مثل هذا العمل؛ فإنه يقوم بإنشاء فرع تنفيذي خاصٍّ للمعاملة بالتعاون مع نظام إدارة المعاملات في الموقع الذي سيقوم بـتنفيذ هذا العمل. ويتمُّ إعطاءُ هذا الفرع التنفيذي للمعاملة معرفاً خاصاً به. بعدئذٍ؛ وباستخدام معرف الفرع التنفيذي الذي أُنشِئ للعمل؛ يتمُّ إرسال العمل، من خلال نظام إدارة الاتصالات، للموقع الذي سيقوم بـتنفيذ العمل. وبنفس الأسلوب، يمكن للعمل الجديد بعد أن يقوم بالتسجيل في نظام إدارة المعاملات في الموقع الذي سيقوم بتنفيذ العمل، بالنيابة عن المعاملة، من أن يبـدأ أعمالاً جديدةً في مواقع أخرى. وبهذه الطريقة تنشأ مجموعةٌ من الأعمال المنفذة في مواقع مختلفة لكلِّ معاملة يمكِن تصوُّرها على هيئة شجرة تنفيذية (ExeCution Tree) للمعاملة، كما هو مُوضَّح في الشكل (9-7)؛ بحيث يكون كلُّ نظام لإدارة المعاملات على علمٍ بجميع نظم إدارة الموارد المشاركين في تنفيذ المعاملة في موقعه؛ بالإضافة إلى نظم إدارة المعاملات المجاورين له في تنفيذ المعاملة (Al-Houmaily, 2014).



شكل رقم (9–7): مثال لإحدى الشجرات التنفيذية للمعاملات.

يُوضِّح الشكل (9-7) مثالاً لشجرةٍ تنفيذيةٍ لمعاملة مصدرها الموقع (1) لنظام قاعدة بـيانات موزَّعة. ويُبيِّن الشكلُ أن المعاملة قد قامت بتنفيذ جزءٍ من عملياتها على نظام إدارة الموارد رقم (1) في الموقع الذي نشأت فيه (الموقع رقم (1))، وامتدَّ تنفيذ عملياتها إلى نظام إدارة الموارد رقم (1) في الموقع رقم (2)، ونظامي إدارة الموارد رقم (1) ورقم (2) في الموقع رقم (3). كما امتدَّ

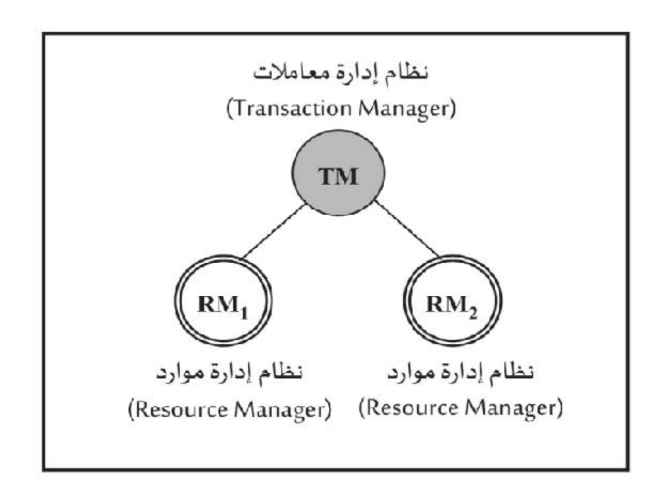
تنفيذ عمليات المعاملة ليشمل نظم إدارة الموارد رقم (1) ورقم (2) ورقم (3) في الموقع رقم (4)، ونظام إدارة الموارد رقم (1) في الموقع رقم (5). وبناءً على ما يُوضِّحه الشكل رقم (9-7)، وعلى ما سبق من شرحٍ للخصائص الوظيفية لنظام إدارة المعاملات؛ فإن نظام إدارة المعاملات في الموقع رقم (1) على علم بمشاركة نظام إدارة الموارد رقم (1) في موقعه، ونظم إدارة المعاملات في الموقعين رقم (2) ورقم (3) في تنفيذ المعاملة. وكذلك هو الحال بالنسبة لنظام إدارة المعاملات، على سبيل المثال، في الموقع رقم (2) الذي هو علم بمشاركة نظام إدارة الموارد رقم (1) في موقعه ونظامي إدارة المعاملات في الموقع رقم (1) والموقع رقم (4). وبحسب هذا النموذج التنفيذي للمعاملات الموزَّعة تنشأ علاقة تنسيقية بين المكونات المشاركة في تنفيذ كل معاملة تُسمَّى «الرئيس والمرؤوس» (Superior/Subordinate) أو «السيد والخادم» (Master/Slave). فعلى سبيل المثال، يُعَدُّ نظام إدارة المعاملات في الموقع رقم (1) في الشكل هو المنسق الرئيسي (Coordinator) للمعاملة؛ في حين تُعَدُّ نظم إدارة المعاملات في الموقع رقم (2) والموقع رقم (3) والموقع رقم (4) والموقع رقم (5) منسقين وسيطين (CasCaded Coordinators). ويُعَدُّ نظام إدارة المعاملات منسقاً وسيطاً عندما يكون مرؤوساً من قِبل نظام لإدارة معاملات وفي نفس الوقت منسقاً لنظم أخرى سواءً أكانت نُظماً لإدارة الموارد فحسب (مثل: نظام إدارة المعاملات في الموقع رقم (4)، ونظام إدارة الموارد في الموقع رقم (5)) أو منسقاً لمزيج من نظم إدارة المعاملات، ونظم إدارة الموارد (مثل: نظام إدارة المعاملات في الموقع رقم (2) ونظم إدارة المعاملات في الموقع رقم (3)). أمَّا نظم إدارة الموارد في المواقع المشاركة كافةً فتُعَدُّ نظماً مشاركة نهائية (Leaf PartiCipants).

بعد انتهاء المعاملة من تنفيذ كافة عملياتها؛ فإنها تقوم بإشعار نظام إدارة المعاملات في الموقع الذي ابتدأت تنفيذها فيه (وهو نظام إدارة المعاملات في الموقع رقم (1) في الشكل رقم (9-7)) بذلك من خلال تعليمة التثبيت (Commit). وعند استقبال نظام إدارة المعاملات لهذه التعليمة؛ فإنه يقوم ببدء تنفيذ بروتوكول خاص لضمان نووية المعاملة على كافة النظم التي شاركت في تنفيذ المعاملة. ويُسمَّى هذا البروتوكول «بروتوكول التثبيت ذو المرحلتين» (Two-Phase Commit (2PC)) (Al-Houmaily, 2018).

9-4-2-1-1 بروتوكول التثبيت ذو المرحلتين (Two-Phase Commit (2PC)):

على الرغم من أن خصائص كل معاملة تُعَدُّ مضمونةً عند تنفيذ المعاملة على واحد فقط من نظم إدارة الموارد؛ فإن خاصية النووية لا يمكن ضمانها عند تنفيذ المعاملة على أكثر من نظام؛ لكونه لا يُوجَد ما يضمن تثبيت نتائج المعاملة على كافة النظم التي شاركت في تنفيذها. ويُعزَى السبب في ذلك إلى الأعطال التي قد تصيب المواقع المختلفة أو تواصلها مع بعضها، وبالتالي تثبيت نتائج المعاملة على بعضٍ من النظم التي شاركت في تنفيذ المعاملة وتراجعها عن التثبيت على النظم الأخرى. فعلى سبيل المثال: قد يرسِل نظامُ المعاملات في الموقع الرئيسي لمعاملةٍ ما تعليمة التثبيت لكافة المواقع التي شاركت في تنفيذ المعاملة؛ بالإضافة إلى نظم إدارة الموارد المشاركة في تنفيذ المعاملة في موقعه؛ غير أن تعليمة التثبيت المُرسَلة قد يتمُّ استقبالها من قِبل نظم إدارة الموارد داخل الموقع، وبعض المواقع الأخرى؛ ولكنها تفشل في الوصول للمواقع المشاركة المتبقية نتيجة عطل في نظام إدارة الاتصالات أو في خطوط شبكة الاتصالات. في هذه الحالة؛ سيتمُّ تثبيت نتائج المعاملة من قبل النظم التي استلمت تعليمة التثبيت؛ ولكنه سيتمُّ إزالة أثر كافة العمليات التي أجرتها المعاملة من قِبل النظم التي لم تستلم تعليمة التثبيت نتيجةً للعطل؛ الأمر الذي سيخلُّ بنووية المعاملة. وللتغلب على هذه المعضلة؛ فإنه لا بد من استخدام بروتوكول خاص؛ لضمان نووية المعاملات في نظم قواعد البيانات الموزَّعة.

إن أكثر هذه البروتوكولات استخداماً لتثبيت نتائج المعاملات في النظم الموزَّعة؛ هو «بروتوكول التثبيت ذو المرحلتين» (Two-Phase Commit (2PC)) (Al-Houmaily, 2010). ولشرح تفاصيل هذا البروتوكول بشكله المبسَّط؛ لنفترض تنفيذ معاملةٍ ما على موقعٍ واحدٍ فقط، ولكن على أكثر من مورد للبيانات في الموقع، كما هو مُوضَّح في الشكل رقم (9-8).

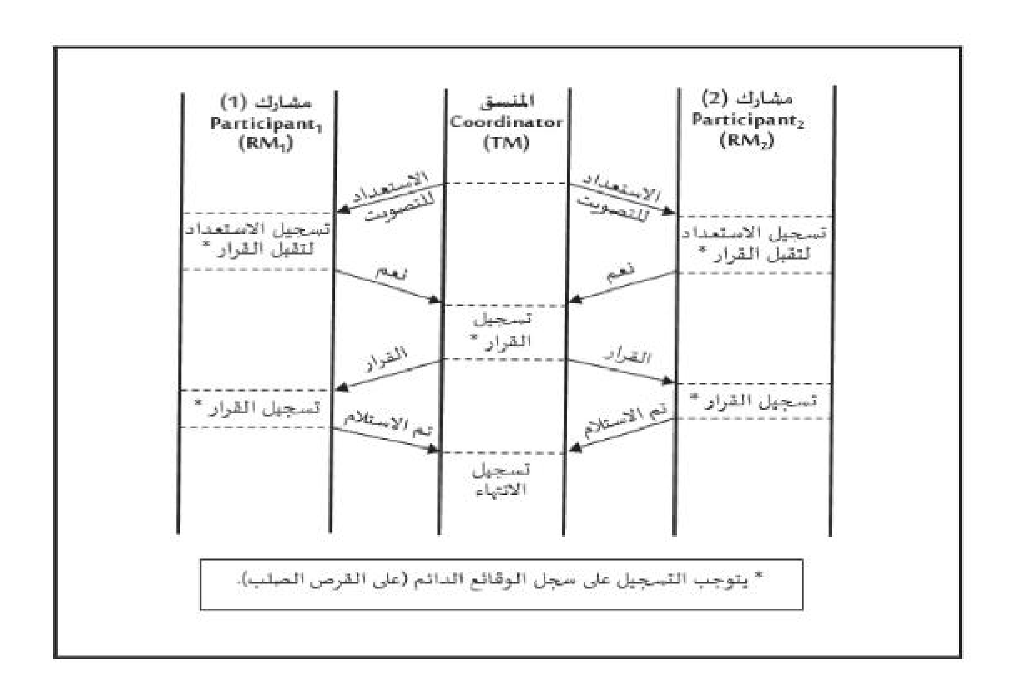


شكل رقم (9–8): مثال لشجرة تنفيذية لمعاملةٍ في موقع واحد.

إن مُسمَّى البروتوكول يدلُّ على طبيعته؛ كونه يتكون من مرحلتين، كما يلي:

1- مرحلة التصويت (Prepare Phase): في بداية هذه المرحلة، وكما هو مُوضَّح في الشكل رقم (9-9)؛ يقوم نظام إدارة المعاملات (وهو منسق عملية تثبيت نتائج المعاملة) بإرسال رسالة يطلب فيها من المشاركين في تنفيذ المعاملة (وهم: نظام إدارة الموارد رقم (1)، ونظام إدارة الموارد رقم (2) في مثالنا الموضَّح في الشكل (9-8)) التصويت عن قدرة كلٍّ منهم على تثبيت نتائج المعاملة في حال أُتخِذ القرار النهائي بالتثبيت. وعند وصول الرسالة لأحد المشاركين في تنفيذ المعاملة؛ فإنه يقوم بالتحقق من قدرته على الالتزام بقرار التثبيت. فإذا كان بمقدوره الالتزام بذلك؛ فإنه يقوم بإرسال رسالة جوابية "نعم". أمَّا إذا لم يكن بمقدوره ذلك؛ فإنه يقوم بإرسال رسالة جوابية "لا".

2- مرحلة اتخاذ القرار (Commit Phase): عند وصول أصوات المشاركين وكان تصويت الجميع يدل على قدرتهم بالالتزام بقرار التثبيت ("نعم")؛ يكون قرار المنسق (وهو نظام إدارة المعاملات) التثبيت (Commit). أمَّا إذا كان صوت أيٍّ من المشاركين "لا"؛ فيكون القرار هو التراجع (Abort). وعند الوصول لقرارٍ بالتثبيت؛ يقوم المنسق بإرسال رسالة تثبيت (Commit) لكلِّ مشارك في تنفيذ المعاملة. أمَّا إذا كان القرار هو التراجع؛ فيقوم المنسِّق بإرسال رسالة تراجع (Abort) للمشاركين الذين صوتوا "نعم". وعند استلام القرار النهائي من قِبل أحد المشاركين؛ يقوم المشارك بتنفيذ القرار (سواءً أكان بالتثبيت أو بالتراجع)، ومن ثم إرسال رسالة جوابية تفيد باستلامه للقرار النهائي وتنفيذه للقرار بشأن المعاملة.



شكل رقم (9–9): بروتوكول التثبيت ذو المرحلتين.

إن قدرة البروتوكول على الوصول لقرار مُوحَّد عبر كافة النظم المشاركة في تنفيذ معاملةٍ ما يتمُّ من خلال تسجيل تقدُّم تنفيذ البروتوكول ضِمن سجل الوقائع الخاص بكلِّ مشارك. وهذه السِّجلات يتوجَّب أن تدوَّن في سجل الوقائع الدائم (على القرص الصلب) ولا يُكتَفى بتدوينها في سجل الوقائع المؤقت في الذاكرة الرئيسية للحاسب الآلي؛ بحيث يتمكَّن المشارك من استرجاع هذه السجلات بعد حدوث أيِّ عطل للحاسب الآلي، ومعرفة الحالة التي وصلت إليها المعاملة قبل العطل. ووفقاً لهذا البروتوكول يتحتَّم على كلِّ مشارك أن يقوم بتسجيل "الاستعداد لتقبل القرار" في سجل وقائعه قبل إرسال صوت "نعم". كما يتحتَّم على المشارك تسجيل "القرار" قبل إرساله لرسالته الجوابية باستلام القرار. أمَّا فيما يتعلق بالمنسق (وهو نظام إدارة المعاملات في الموقع الذي ابتدأت المعاملة تنفيذها فيه)؛ فإنه يتوجَّب عليه تسجيلُ سجل "القرار" قبل إرسال القرار للمشاركين في تنفيذ المعاملة. أمَّا سجل "الانتهاء"؛ فيُكتفى بتسجيله ضِمن سجل الوقائع في الذاكرة الرئيسية، ولا يتحتَّم تمريره لسجل الوقائع الدائم (على القرص الصلب). والسبب في ذلك أنه بإمكان المنسق، وفقاً لهذا البروتوكول، من التثبت من وصول القرار لكافة المشاركين دون الحاجة إلى تسجيل هذا السجل على سجل الوقائع الدائم. كما أن هذا السجل لا يؤثر على صحة البروتوكول (الذي يقضي بالتوصل لقرار مُوحَّد عبر كافة المشاركين في تنفيذ المعاملة)، وإنما

يُستخدَم من قِبل المنسق للتعرُّف على المعاملات التي انتهى تنفيذها بالكامل - عبر كافة المشاركين - وإتلاف سجلاتها من سجل الوقائع الدائم (لتنظيفه من المعاملات المنتهية وتقليل حجمه) في مراحل لاحقة، إن استدعى الأمر ذلك.

الجديرُ بالذكر أن التسجيل في سجل الوقائع الدائم يُعَدُّ مُكلِّفاً جداً مقارنةً بالتسجيل في سجل الوقائع في الذاكرة الرئيسية (من حيث الوقت الزمني)؛ كونه يتطلَّب الرجوع للقرص الصلب الذي يرتبط بحركة ميكانيكية للقرص عوضاً عن العملية الكهربائية التي ترتبط بـها عملية التسجيل في سجل الوقائع في الذاكرة الرئيسية. ويعني هذا أنه كلما ازداد عددُ السجلات الواجب تدوينها في سجل الوقائع الدائم من قِبل بروتوكول التثبيت؛ أدَّى ذلك إلى انحدار أداء النظام (من خلال ملاحظة الوقت اللازم لتنفيذ المعاملات عليه).

9-4-2-1-1-1 التشافي من الأعطال عند استخدام «بروتوكول التثبيت ذو المرحلتين»:

يتمُّ التعرُّف على أعطال التواصل بين المواقع المختلفة باستخدام طريقة الوقت المُستَقطَع (Timeouts). وقد يكون عطل التواصل نتيجةً لعطلٍ في خطوط الاتصالات (CommuniCation Failure)، أو عطل لأحد المواقع (Site Failure).

يُوجَد في بروتوكول التثبيت ذي المرحلتين أربعُ حالاتٍ من الممكن أن يحصل فيها عطلٌ للتواصل؛ بسبب خطوط الاتصال. الحالة الأولى عندما يكون أحد المشاركين بـانتظار تعليمة جديدة من المعاملة لتنفيذها، أو وصول رسالة ‘‘الاستعداد للتصويت’’ من المنسِّق. في هذه الحالة من الممكِن للمشارك - كونه لم يعطِ للمنسِّق أيَّ وعدٍ بخصوص استعداده لتثبيت نتائج المعاملة - أن يتراجع عن تنفيذ المعاملة بمجرد تعرُّفه على هذا العطل. أما الحالة الثانية؛ فقد تحدث عندما يكون المنسِّق قد أرسل رسائل استعداد التصويت ولكن قبل أن تصله كافة الأصوات. في هذه الحالة من الممكن للمنسِّق - كونه لم يتخذ قراراً بتثبيت نتائج المعاملة، وبالتالي عدم تمكُّن أيٍّ من المشاركين من تثبيت نتائج المعاملة - أن يتراجع عن تثبيت المعاملة. الحالة الثالثة هي عندما يكون أحدُ المشاركين قد أرسل صوته بالالتزام بالقرار النهائي حيال المعاملة ولكن قبل أن يصله القرار النهائي من قِبل المنسِّق. في هذه الحالة يكون المشارك في وَضعٍ مُعَلَّق لا يسمح له بتثبيت نتائج المعاملة أو التراجع عنها حتى يتم إصلاح العطل. وبعد إصلاح العطل؛ يقوم المشارك بالاستفسار من المنسِّق عما آل إليه وَضْع المعاملة من حيث القرار النهائي بشأنها. وحيث إنه ليس من الممكن

أن يكون المنسق قد نسي النتيجة النهائية للمعاملة - كونه لم يستقبل رسائل إتمام الاستلام كافة - فإنه سيقوم بتزويد المشارك بـالقرار النهائي للمعاملة. الحالة الرابعة عندما يكون المنسِّق بانتظار رسائل إتمام الاستلام من قِبل المشاركين. في هذه الحالة يقوم المنسق - كونه غير متأكدٍ من استلام جميع المشاركين لقراره، وبالتالي نسيان المعاملة أو مَسْح بياناتها من سجل وقائعه عند الضرورة - بإعادة إرسال القرار للمشاركين الذين انقطع التواصل معهم بمجرد إصلاح العطل متوقعاً منهم رسائل إتمام الاستلام.

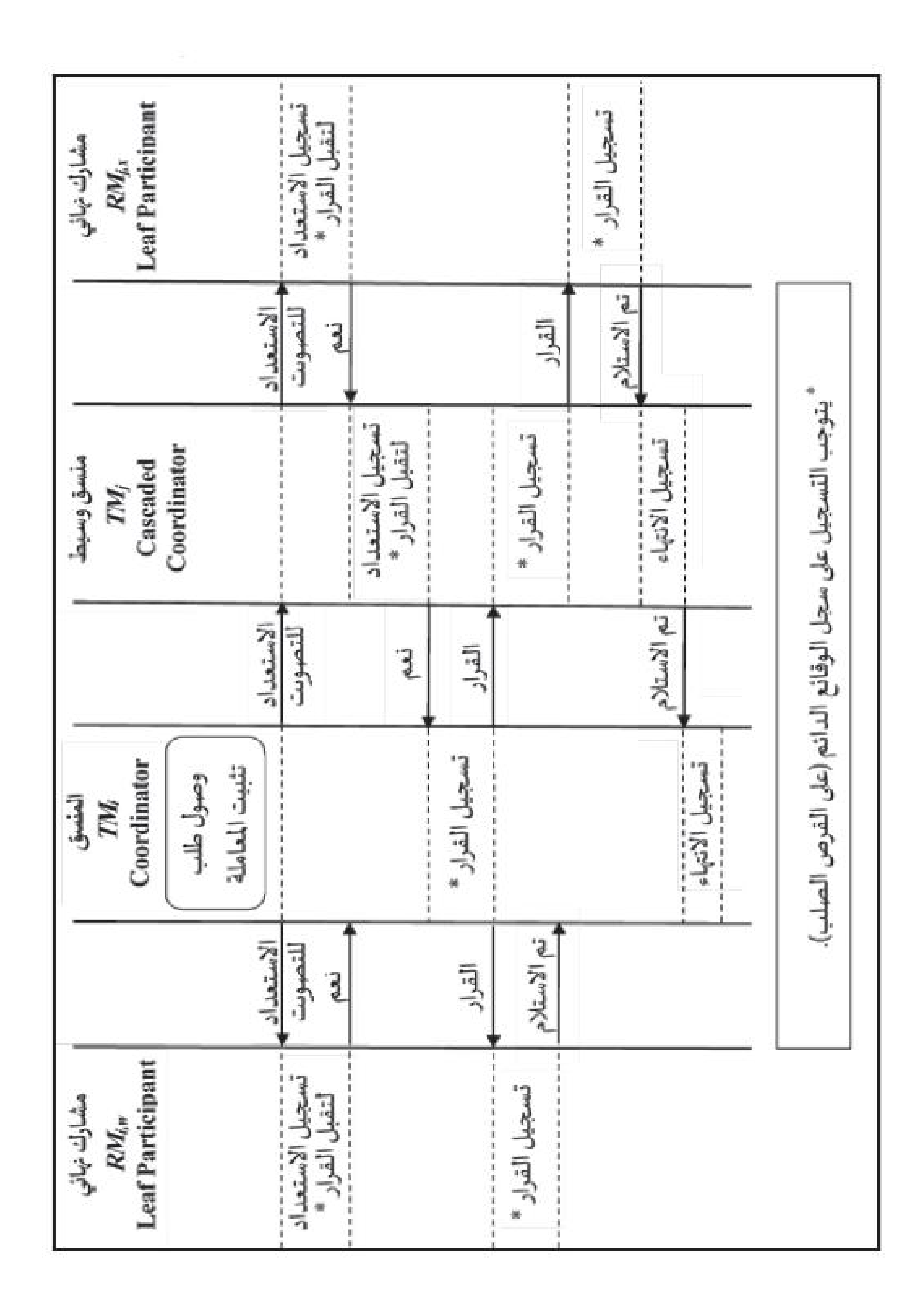
للتشافي من أعطال المواقع؛ هنالك حالتان للتعامل معهما، وهما: عطل موقع المنسِّق، وعطل موقع أحد المشاركين. عند تعطل المنسِّق، يقوع المنسِّق ببناء نظام البروتوكول لديه من خلال مسح سجل وقائعه الدائم؛ بحيث يقوم بإدراج ما وصلت إليه كلُّ معاملة ابتدأت تنفيذ البروتوكول قبل العطل؛ ولكنها لم تتمكَّن من إنهائه بسبب العطل. ويعني هذا أنه على المنسِّق إدراج كل معاملة لديها قرار نهائي (بالتثبيت أو بالتراجع) ضِمن سجل الوقائع الخاص بالمنسق؛ ولكن دون ما يقابله من سجل يُبيِّن انتهاءها من البـروتوكول (سجل الانتهاء). بعد ذلك؛ يقوم المنسِّق بإرسال قراره النهائي - حسب ما جاء في سجل وقائعه - لكلِّ مشارك من المشاركين في إحدى هذه المعاملات، متوقعاً ردّ كلِّ مشارك. وعندما تصل الرسالة لأحد المشاركين في إحدى هذه المعاملات؛ يقوم المشارك بتنفيذ قرار المنسِّق ومن ثم الرَّد برسالة إتمام الاستلام في حالِ لم يستلم المشاركُ هذا القرار قبل حدوث العطل وكان معلقاً، أو أن يقوم فقط بالرَّد برسالة إتمام الاستلام في حال كونه قد استلم القرار قبل العطل وقام بتنفيذه وأنهى المعاملة لديه ولم يَعُد يذكرها.

عند تعطُّل موقع أحد المشاركين؛ يقوم المشاركُ ببناء نظام البروتوكول لديه من خلال مَسْح سجل وقائعه - كما هو الحال عند تعطل المنسق - بحيث يقوم بإدراج ما وصلت إليه كلُّ معاملة ابتدأت تنفيذ البروتوكول قبل العطل؛ ولكنها لم تتمكَّن من إنهائه بسبب العطل. ولكن في هذه الحالة على المشارك إدراج كلِّ معاملة لديها سجل يدلُّ على تصويت المشارك بالإذعان للقرار النهائي "نعم"؛ ولكن دون ما يقابله من سجل يبيِّن انتهاءها من البـروتوكول (سجل القرار). بعد ذلك؛ يقوم المشارك بـإرسال رسالة استفسار عن القرار النهائي لمنسِّق كل واحدة من هذه المعاملات. وحيث إنه ليس من الممكن أن يكون منسِّق أيٍّ من هذه المعاملات قد نسي قراره النهائي بشأن المعاملة -

لكونه لم يستلم رسائل إتمام الاستلام كافة - فإن المنسق سيقوم بالرَّد برسالة تتضمَّن قراره النهائي تجاه المعاملة. وبمجرد وصول القرار؛ يقوم المشارك بتنفيذه، ومن ثم، الرَّد برسالة إتمام الاستلام.

9-4-2-1-1-2 بروتوكول التثبيت ذو المرحلتين المتعَدِّد المستويات:

يُوضِّح الشكل رقم (9-10) إحدى المعاملات التي قامت بتنفيذ عملياتها على موقعين، وهما: الموقع (i) (Sitei)، والموقع (j) (Sitej). كما يوضِّح الشكلُ أن جزءاً من عمليات المعاملة قد تمَّ تنفيذه من قِبل أحد نظم إدارة الموارد في موقعها الأصلي، وهو: RMi,w، والجزء الآخر قد تمَّ تنفيذه على أحد نظم إدارة الموارد في الموقع الآخر، وهو: RMj,x. وبذلك يصبح نظام إدارة المعاملات في الموقع الذي نشأت فيه المعاملة (TMi) هو المنسِّق الرئيس لإنهاء تنفيذ المعاملة؛ في حين يصبح نظام إدارة المعاملات في الموقع الثاني (TMj) منسقاً وسيطاً.



شكل رقم (10-9): بروتوكول التثبيت ذو المرحلتين متعدد المستويات.

عند انتهاء المعاملة من تنفيذ عملياتها كافةً وإرسالها لعبارة تثبيت النتائج للمنسِّق؛ يقوم المنسق ببدء تنفيذ بروتوكول التثبيت ذي المرحلتين؛ من خلال إرساله لرسالة «الاستعداد للتصويت»؛ وذلك لكلٍّ من نظام إدارة الموارد الذي شارك في تنفيذ المعاملة في موقعه (RMi,w)، ولنظام إدارة المعاملات في الموقع الثاني (TMj)؛ كونه المنسِّق لنظم إدارة الموارد في ذلك الموقع. وكما يُلاحَظ في الشكل؛ فإن تسلسل الرسائل وتسجيل الوقائع بالنسبة للمنسِّق الرئيسي للمعاملة (TMi)، ولكلِّ مشارك نهائي (وهما: نظام إدارة الموارد في الموقع الأول (RMi,w)، ونظام إدارة الموارد في الموقع الثاني (RMj,x))؛ فإنها تأتي متطابقةً مع بروتوكول التثبيت ذي المرحلتين الذي سبق شرحه في الجزء السابق. والفرق بين بروتوكول التثبيت ذي المرحلتين المتعَدِّد المستويات، وسابقه ذي المستوى الواحد - يكمُن في سلوك كلِّ منسِّق وسيط. فالمنسِّق الوسيط (وهو نظام إدارة المعاملات في الموقع الثاني لتنفيذ المعاملة في مثالنا (TMj)) يتعامل مع المنسِّق الذي يعلوه في الشجرة التنفيذية للمعاملة، وكأنه مشارك نهائي (من حيث الرسائل التي يرسلها له)، وفي نفس الوقت، يتعامل مع مَنْ يدنوه من نظم في الشجرة التنفيذية للمعاملة (سواءً أكانوا نظماً لإدارة موارد في موقعه، كما هو حال (RMi,w) في مثالنا، أو أكانوا نظماً لإدارة معاملات في مواقع أخرى، كما هو حال (TMj))، كمنسِّق رئيسي (من حيث الرسائل التي يرسلها لهم).

يقوم المنسق الوسيط عند وصول رسالة «الاستعداد للتصويت»، من المنسِّق الذي يعلوه في الشجرة التنفيذية للمعاملة، بتوجيه الرسالة لكلِّ مشارك يدنوه في الشجرة. ويقوم بعد ذلك بانتظار وصول رسائل التصويت الجوابية من المشاركين (كما هو مُوضَّح في الشكل (9-10)). وعند وصول رسائل التصويت على المعاملة، وعلى افتراض أن التصويت كان «نعم» من الجميع؛ يقوم المنسِّق الوسيط بتسجيل استعداده لتقبل القرار في سجل الوقائع (الدائم) الخاص به على القرص الصلب؛ ومن ثم إرسال رسالته التصويتية «نعم» للمنسِّق الذي يعلوه في الشجرة. أمَّا في حال كان تصويت أيٍّ من المشاركين «لا»؛ فيقوم المنسق الوسيط بإنهاء المعاملة لدية على أساس التراجع عن تنفيذ المعاملة. وعليه؛ يقوم المنسِّق الوسيط بإرسال رسالة قرار «بالتراجع» (Abort) لكلِّ مشارك قام بالتصويت «نعم»، ورسالة تصويت «لا» للمنسِّق الذي يعلوه في الشجرة التنفيذية

للمعاملة. ويعكس تصويت المنسِّق الوسيط، في كلا الحالتين، تصويته الخاص بالإضافة لتصويت جميع المشاركين الذين يدنونه في الشجرة التنفيذية للمعاملة.

وعندما تصل رسالة بالقرار النهائي إلى منسِّق وسيط ما من المنسِّق الذي يعلوه في الشجرة التنفيذية للمعاملة؛ يقوم المنسِّق الوسيط بتسجيلها في سجل الوقائع الدائم لديه وتنفيذ القرار؛ ومن ثم يقوم بتوجيهها للمشاركين الذين يدنونه في الشجرة التنفيذية للمعاملة. وعند وصول رسائل المشاركين الدالة على استلامهم للقرار النهائي وتنفيذه؛ يقوم المنسق الوسيط بتسجيل «الانتهاء» من المعاملة في سجل وقائعه المخزَّن في الذاكرة الرئيسية (دون الحاجة إلى تمريره لسجل الوقائع الدائم على القرص الصلب)، ومن ثم إرسال رسالة تفيد باستلام القرار النهائي للمنسِّق الذي يعلوه في الشجرة التنفيذية للمعاملة.

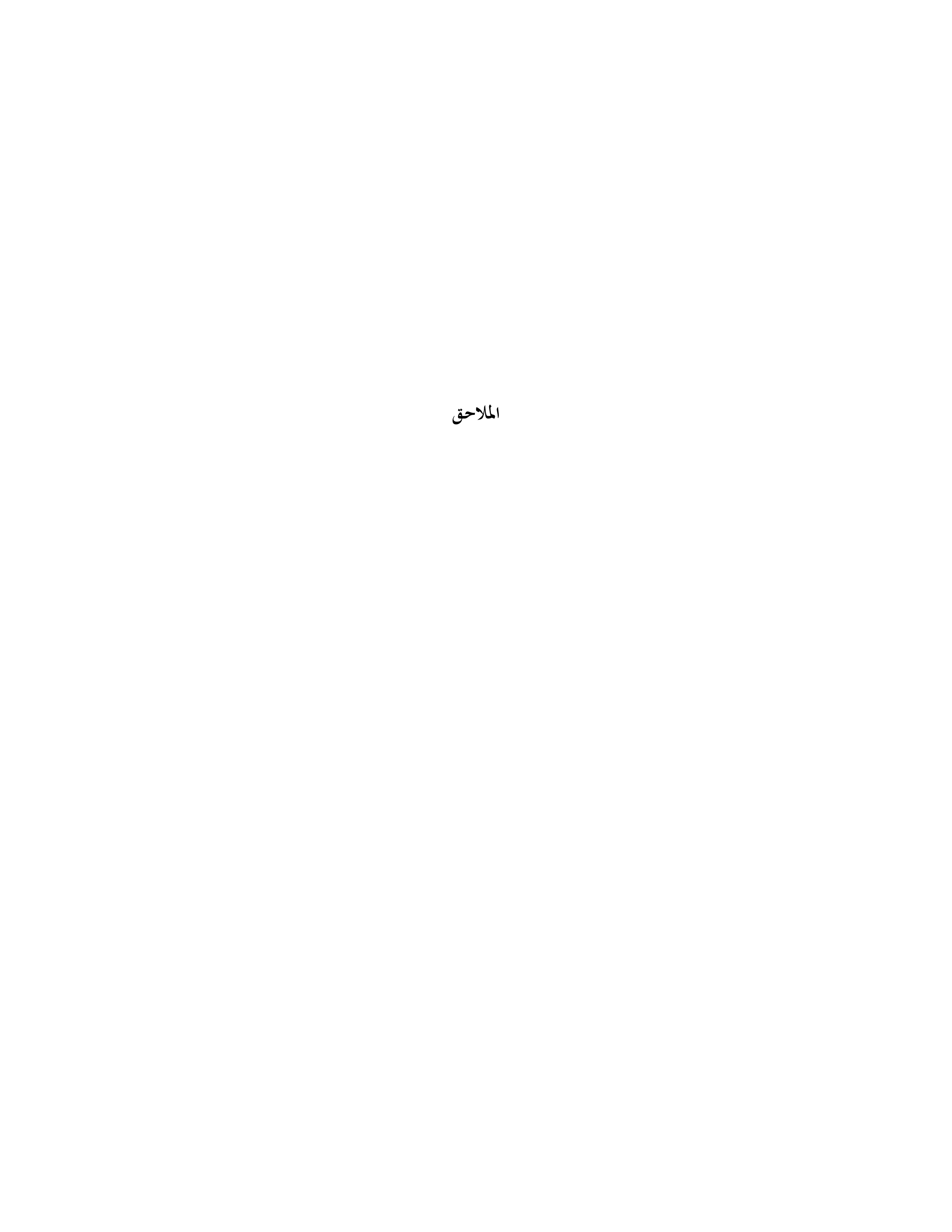
9-4-2-1-1-2-1 التشافي عند استخدام «بروتوكول التثبيت ذو المرحلتين المتعَدِّد المستويات»:

إن عملية التشافي من الأعطال عند استخدام بروتوكول التثبيت ذي المرحلتين المتعَدِّد المستويات؛ شبيهةٌ إلى حَدٍّ كبيرٍ مع بروتوكول التثبيت ذي المرحلتين. فعملية التشافي من أعطال الاتصالات وأعطال المواقع التي قد تصيب كلَّ مشاركٍ نهائي، وأيَّ منسقٍ رئيسي - تأتي متطابقةً مع بروتوكول التثبيت ذي المرحلتين. والفرق الوحيد بين البروتوكولين؛ هو وجود المنسِّق الوسيط في بروتوكول التثبيت ذي المرحلتين المتعدد المستويات، وكيفية تشافيه من أعطال الاتصالات وأعطال المواقع التي قد تصيبه.

يُوجَد في بروتوكول التثبيت ذي المرحلتين المتعَدِّد المستويات؛ أربعُ حالاتٍ من الممكن أن يحصل فيها عطل لتواصل المنسِّق الوسيط مع المنسِّق الذي يعلوه أو المشاركين الذين يدنونه في الشجرة التنفيذية للمعاملة بسبب خطوط الاتصال. الحالة الأولى عندما يكون المنسِّق الوسيط بانتظار رسالة «الاستعداد للتصويت» من قِبل المنسِّق الذي يعلوه في الشجرة التنفيذية؛ إيذاناً بـانتهاء المعاملة. في هذه الحالة يقوم المنسِّق بالتعامل مع المعاملة وكأنها قد قامت بـالتراجع. وبناءً عليه؛ يقوم المنسِّق الوسيط بتنفيذ عملية التراجع لديه والقيام بإرسال رسالة «تراجع» لكلٍّ من المشاركين الذين يدنونه في الشجرة التنفيذية للمعاملة. الحالة الثانية قد تحدث عندما يتعذر وصول تصويت واحدٍ أو أكثر من المشاركين الذين يدنون المنسِّق الوسيط في الشجرة التنفيذية للمعاملة بعد أن قام المنسِّق الوسيط بإرسال رسائل «الاستعداد للتصويت» طالباً تصويتهم. في هذه الحالة؛

يقوم المنسِّق الوسيط، وكما هو الوضع في الحالة السابقة، بالتعامل مع المعاملة وكأنها قد قامت بـعملية تراجع. وبناءً عليه، يقوم المنسِّق الوسيط بـإرسال رسالة «لا» للمنسِّق الذي يعلوه في الشجرة التنفيذية، ورسالة قرار «تراجع» لكلِّ مشارك يدنوه فيها. الحالة الثالثة قد تحدث بعد إرسال المنسِّق الوسيط رسالة «نعم» للمنسِّق الذي يعلوه وقبل وصول رسالة «القرار النهائي». في هذه الحالة يبقى المنسِّق الوسيط في حالة معلقة لا يستطيع من خلالها البتُّ في حالة المعاملة إلى حين إصلاح العطل. وبعد إصلاح العطل؛ يقوم المنسق الوسيط بالاستفسار عن القرار النهائي حيال المعاملة من المنسِّق الذي يعلوه في الشجرة التنفيذية، والقيام بـتنفيذ القرار حال وصوله. الحالة الرابعة قد تحدث بعد إرسال القرار النهائي للمشاركين الذين يدنون المنسِّق الوسيط في الشجرة التنفيذية للمعاملة؛ ولكن قبل وصول رسائلهم الجوابية كافةً. في هذه الحالة؛ لن يتمكَّن المنسِّق الوسيط من مواصلة الـبروتوكول، ويتوجَّب عليه الانتظار إلى حين إصلاح العطل. وبعد إصلاح العطل؛ يقوم المنسق الوسيط بإعادة إرسال القرار النهائي لكلِّ مشارك تعَذَّر وصول رسالته الجوابية إزاء القرار النهائي. وعند وصول الرسائل المطلوبة؛ يواصل المنسِّق الوسيط استكمال البروتوكول.

أمَّا في حال تعطل موقع المنسق الوسيط عن العمل؛ يقوم المنسِّق الوسيط، بعد إصلاح العطل وعند إعادة تشغيله، بمسح سجل الوقائع الدائم لديه للتعرُّف على كلِّ معاملة ابتدأت تنفيذ البروتوكول لديه؛ ولكنها لم تنتهِ من مراحله كافةً. والمعاملات المقصودة هنا هي تلك المعاملات التي يُوجَد لكلٍّ منها سجل يدلُّ على «الاستعداد لتقبل قرار التصويت» دون ما يقابله من سِجل يدل على «القرار»، أو سجل يدلُّ على « الاستعداد لتقبل قرار التصويت» وسجل مقابل له يدل على «القرار»؛ ولكن دون سجل يدل على «الانتهاء». ففي الحالة الأولى؛ يقوم المنسِّق الوسيط بـالاستفسار عن القرار النهائي للمعاملة من المنسِّق الذي يعلوه في الشجرة التنفيذية للمعاملة. وبمجرد وصول رسالة «القرار النهائي»، يواصل المنسِّق الوسيط استكمال البـروتوكول. أمَّا في الحالة الثانية؛ يقوم المنسِّق الوسيط بإرسال رسالة «القرار النهائي» لكلِّ مشارك يدنوه في الشجرة التنفيذية للمعاملة؛ وذلك حتى يتمكَّن المنسق الوسيط من التأكُّد من وصول القرار لكافة المشاركين الذين يدنونه والقيام بتسجيل سجل «الانتهاء».



الملاحق

ملحق رقم (1)

حالة دراسية – قاعدة بيانات جامعة أهلية افتراضية

ملحق رقم (1) –1 قواعد العمل المعمول بها في الجامعة:

تنفذ الجامعة الأهلية مجموعةً من المواد الدراسية في كلِّ فصل دراسي. ومن قواعد العمل المتبعة في الجامعة الأهلية، ما يلي:

1- يُوجَد في الجامعة عددٌ من الأقسام الدراسية، ولكل قسم (Department) من الأقسام العلمية رمز (Department_ID) يميِّزه عن بقية الأقسام، واسم (Name).

2- يعمل في الجامعة عددٌ من أعضاء هيئة التدريس، ولكلِّ عضو هيئة تدريس (FaCulty) رمز (FaCulty_ID) يميِّزه عن بقية أعضاء هيئة التدريس، واسم (Name) يتكون من (الاسم الأول (FName) واسم العائلة (LName))، وراتب شهري (Salary)، وتاريخ ميلاد (Date of Birth (DOB))، ورقم هاتف (Phone_No).

3- يدرس في الجامعة عددٌ من الطلاب، ولكلِّ طالب (Student) رمز (Student_ID) يميِّزه عن بقية الطلاب في الجامعة، واسم (Name) يتكون من (الاسم الأول (FName) واسم العائلة (LName))، وعنوان بريدي (Address) يتكوَّن من (اسم الشارع (Street)، واسم المدينة (City)، والرمز البريدي (Zip_Code)).

4- تُنفِّذ الجامعة مجموعةً من المواد الدراسية، ولكلِّ مادة دراسية (Course) رمز (Course_ID) يميِّزها عن بقية المواد الدراسية التي تنفذها الجامعة، واسم (Title)، وعدد وحدات (أو ساعات) دراسية (Units).

5- تُنفَّذ (أو تعقد) كلُّ مادة دراسية من خلال مجموعة (أو شعبة) دراسية (SeCtion) واحدة أو أكثر في الفصل الدراسي الواحد، أو قد لا تنفذ (أو تعقد) أيَّة مجموعة (أو شعبة) للمادة الدراسية في فصل دراسي معين، ولكل مجموعة دراسية رمز (SeCtion_ID) يتكون من (رقم المجموعة، والفصل الدراسي المنفذة فيه (Semester)، والسنة الدراسية المنفذة فيها (Year)). أمَّا رقم المجموعة (SeCtion_No)؛ فهو رقم (مثل: 1، 2، 3 ... إلخ) يميِّز المجموعة عن بقية المجموعات المنفِّذة للمادة الدراسية نفسها (وفي نفس الفصل والسنة الدراسيين)؛ ولكنه لا يميِّزها بشكلٍ منفرد عن بقية المجموعات الدراسية المنفذة للمواد الدراسية الأخرى في الجامعة. كما أن لكلِّ مجموعة دراسية موقع من مواقع الجامعة لتنفيذها فيه (LoCation).

6- قد يكون للمادة الدراسية الواحدة مجموعةٌ من المتطلبات الدراسية، أو قد لا يكون للمادة الدراسية أيَّة متطلبات دراسية. كما أن المادة الدراسية الواحدة قد تكون متطلباً لأكثر من مادة دراسية أو قد لا تكون متطلباً لأية مادة دراسية.

7- يعمل في (works for) كلُّ قسمٍ من أقسام الجامعة عضو هيئة تدريس واحد أو أكثر، وكل عضو من أعضاء هيئة التدريس يعمل في قسم دراسي واحد فقط.

8- كلُّ عضو هيئة تدريس في الجامعة مؤهل (Qualified) لتدريس مادة دراسية واحدة على الأقل، وقد يتوفر للمادة الدراسية الواحدة أكثر من عضو هيئة تدريس مؤهلٍ لتدريسها، وقد لا يوجد من أعضاء هيئة التدريس مَنْ هو مؤهل لتدريس المادة.

9- عندما يتأهل عضو هيئة التدريس لتدريس مادة ما لأول مرة؛ يكون هنالك تاريخ لتأهيله (QualifiCation Date) يحَدِّد تاريخ تأهل عضو هيئة التدريس لتدريس المادة الدراسية.

10- تُدَار (Administered) كلُّ مادة دراسية من قِبل قسم دراسي واحد من أقسام الجامعة، ويُدِير كلُّ قسم مادةً دراسيةً واحدةً على الأقل.

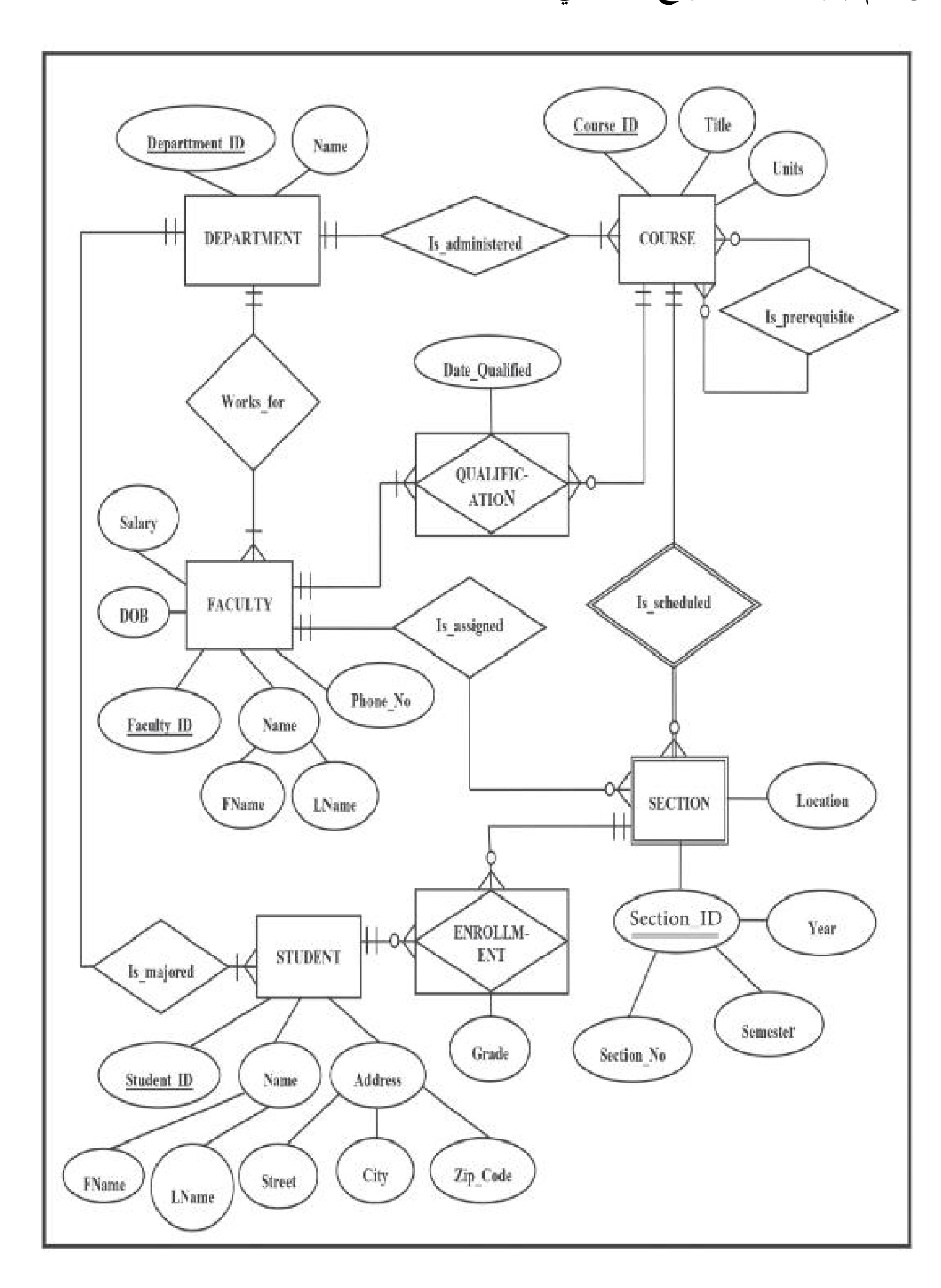
11- قد يُسجِّل (Enrolls) الطالب الواحد في أكثر من مجموعة (أو شعبة) دراسية، أو قد لا يسجل في أية مجموعة (أو شعبة) دراسية، والمجموعة (أو الشعبة) الدراسية الواحدة قد لا يُسجّل فيها أيُّ طالب أو قد يُسجّل فيها أكثر من طالب.

12- عندما يُسجَّل طالب في مجموعة دراسية؛ تكون له درجة (Grade) تُعطَى عند انتهائه من الدراسة في المجموعة.

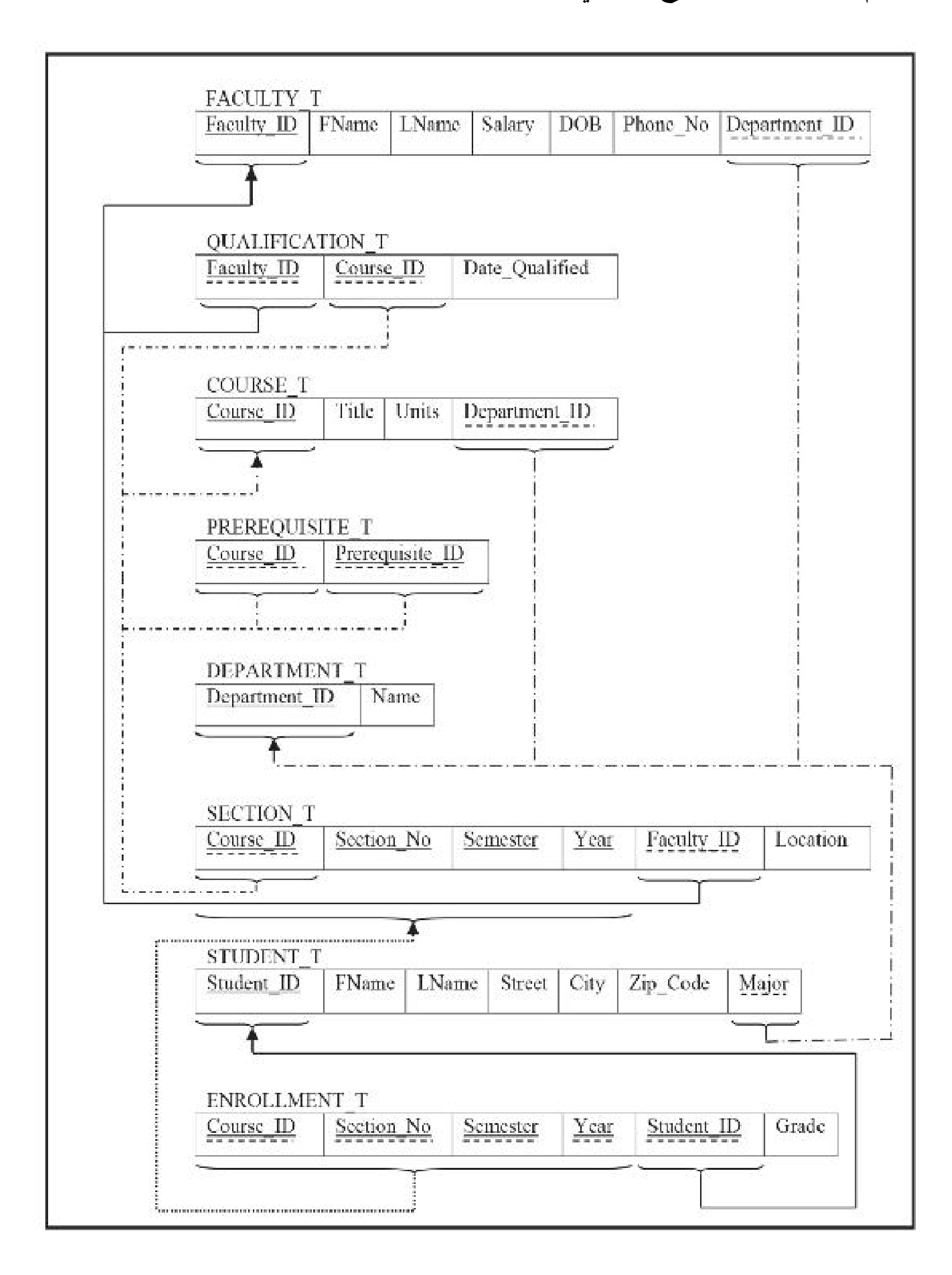
13- يتخصَّص كلٌّ طالب (Majors) في قسمٍ دراسي واحد فقط، ويتخصَّص في القسم الدراسي الواحد أكثر من طالب.

14- يُكَلَّف (Assigned) كلٌّ عضو هيئة تدريس بتدريس مجموعة (أو شعبة) دراسية واحدة أو أكثر، وقد لا يُكلَّف عضو هيئة التدريس بأية مجموعة (أو شعبة) دراسية، والمجموعة (أو الشعبة) الدراسية الواحدة تُكلَّف لعضو هيئة تدريس واحد فقط.

ملحق رقم (1) – 2: النموذج المفاهيمي لقاعدة البيانات:



ملحق رقم (1) – 3: النموذج المنطقي لقاعدة البيانات:



ملحق رقم (1) – 4: جداول قاعدة البيانات حسب بنائها باستخدام نظام إدارة قاعدة بيانات أكسس:

– جدول الأقسام العلمية (DEPARTMENT_T):

DEPARTMENT_T

Department_ID	Name
CHEM	Chemistry
CS	Computer Science
EE	Electrical Engineering
ENGL	English Language
MATH	Mathematics
PHYS	Physics
STAT	Statistics

– جدول المواد الدراسية (COURSE_T):

COURSE_T

Course_ID	Title	Units	Department_ID
CHEM101	Chemistry (I)	3	CHEM
CHEM102	Chemistry (II)	3	CHEM
CS101	Java Programming	3	CS
CS102	Software Engineering	3	CS
CS103	C/C++ Programming	3	CS
CS104	Computer Architecture	3	CS
CS105	Introduction to Database Systems	3	CS
EE101	Electric Circuits	3	EE
EE102	Electronics (I)	3	EE
EE103	Electronics (II)	3	EE
EE104	Communication Networks	4	EE

COURSE_T

Course_ID	Title	Units	Department_ID
ENGL101	English Grammar	2	ENGL
ENGL102	English Writing	3	ENGL
ENGL103	Technical Writing	3	ENGL
MATH101	Introduction to Mathematics	3	MATH
MATH102	Differential Equations	3	MATH
MATH103	Calculus (I)	3	MATH
MATH104	Calculus (II)	3	MATH
MATH106	Algebra	4	MATH
MATH107	Computer Mathematics	3	MATH
PHYS101	Physics (I)	3	PHYS
PHYS102	Physics (II)	3	PHYS
STAT101	Introduction to Statistics	3	STAT
STAT102	Advanced Statistics	3	STAT

– جدول المواد الدراسية المتطلبة (**PRERequisite_T**):

PREREQUISITE_T

Course_ID	Prerequisite_ID
CHEM102	CHEM101
CS102	MATH101
CS103	CS102
CS105	MATH101
EE102	EE101
EE103	EE102
EE103	MATH101
MATH102	MATH101
MATH103	MATH101
MATH104	MATH103

PREREQUISITE_T

Course_ID	Prerequisite_ID
MATH106	MATH101
MATH107	MATH101
PHYS102	PHYS101
STAT102	STAT101

– جدول أعضاء هيئة التدريس (FACULTY_T):

FACULTY_T

Faculty_ID	FName	LName	Phone_NO	Salary	DOB	Department_ID
200	Khalid	Aloufi	454-2341	35000	22/05/1963	MATH
220	Fahad	Alhamid	456-7733	25900	07/10/1970	MATH
310	Saleh	Aleesa	454-8932	30000	13/09/1966	CS
320	Mohammed	Alhamad	454-5412	44000	13/05/1965	CS
330	Ghanim	Alghanim	456-2234	44500	12/08/1969	CS
340	Ibraheem	Alsaleh	454-1234	25000	20/01/1970	CS
400	Ahmad	Alotaibi	454-4563	33900	17/05/1971	CHEM
420	Saleh	Alghamdi	454-2233	44600	13/02/1969	CHEM
500	Yahya	Khorshid	456-2221	36700	12/03/1965	ENGL
540	Salem	Alhamad	456-3304	40000	11/09/1972	ENGL
560	Salman	Albassam	454-7865	33800	13/09/1968	ENGL
600	Turki	Alturki	456-7891	27800	23/07/1975	STAT
640	Fahad	Alzaid	456-3322	44300	12/05/1971	STAT
660	Saud	Alkhalifa	454-9856	44900	13/08/1972	STAT

FACULTY_T

Faculty_ID	FName	LName	Phone_NO	Salary	DOB	Department_ID
710	Mahmood	Alsalem	456-3323	31900	19/02/1973	PHYS
730	Mishal	Almazid	454-2343	29800	17/09/1975	PHYS
770	Sultan	Aljasir	456-3212	43300	13/05/1970	PHYS
800	Ali	Albader	456-7812	45300	22/06/1966	EE
810	Saad	Alzhrani	454-5578	44200	17/10/1967	EE
850	Ahmad	Alsabti	456-0120	33900	15/04/1973	EE

– جدول القدرات التعليمية (QUALIFICATION_T):

QUALIFICATION_T

Couse_ID	Faculty_ID	Date_Qualified
CHEM101	400	02/01/1991
CHEM102	420	02/07/1992
CS101	310	05/06/1995
CS102	320	09/08/1995
CS103	320	03/09/1996
CS104	330	02/10/1997
CS105	340	02/12/1997
EE101	800	08/01/1993
EE102	810	12/03/1994
EE103	850	15/11/1995
EE104	810	03/02/1996
ENGL101	500	01/07/1995
ENGL102	540	02/08/1994
ENGL103	560	09/09/1993
MATH101	200	13/11/1991
MATH102	200	02/06/1993
MATH103	220	02/07/1993

QUALIFICATION_T

Couse_ID	Faculty_ID	Date_Qualified
MATH104	220	13/08/1993
MATH106	220	17/10/1994
MATH107	200	10/01/1995
PHYS101	710	13/07/1996
PHYS101	770	11/02/1996
PHYS102	730	02/01/1997
STAT101	600	15/08/1993
STAT101	660	03/04/1995
STAT102	640	02/05/1994

– جدول المجموعات (أو الشُّعَب) الدراسية (SECTION_T):

SECTION_T

Course_ID	Section_No	Semester	Year	Faculty_ID
CHEM101	1	FALL	2000	400
CHEM101	2	FALL	2000	400
CS101	1	FALL	2000	310
CS101	2	FALL	2000	310
CS102	1	SPRING	2000	320
CS103	1	SPRING	2000	320
CS104	1	FALL	2001	330
CS105	1	SPRING	2001	340
EE101	1	FALL	2001	800
EE102	1	SPRING	2001	810
ENGL101	1	FALL	2000	500
ENGL102	1	SPRING	2000	540
MATH101	1	FALL	2000	200
MATH102	1	SPRING	2000	200

SECTION_T

Course_ID	Section_No	Semester	Year	Faculty_ID
MATH103	1	FALL	2001	220
MATH104	1	SPRING	2001	220
PHYS101	1	FALL	2001	710
PHYS102	1	SPRING	2001	730
STAT101	1	SPRING	2000	600
STAT102	1	SPRING	2001	640

- جدول الطلاب (STUDENT_T):

STUDENT_T

Student_ID	FName	LName	Street	City	Zip_Code	Major
19992020	Saleh	Alhamad	13 Almutanabi Street	Riyadh	11121	CS
19992341	Abdullah	Aloufi	25 Jareer Street	Riyadh	12123	CHEM
19994512	Salem	Algamdi	98 Bin Taimiah Street	Jeddah	34565	PHYS
20001111	Mishal	Alyousef	13 Alsouk Street	Taif	67156	CS
20001212	Khalid	Alsultan	22 Bin Hamdan Street	Jeddah	34565	MATH
20001213	Mohammed	Abdelaleem	10 Bin Hamdan Street	Jeddah	35787	STAT
20001214	Sami	Aloutaibi	67 Alfadel Street	Dammam	26123	ENGL
20001215	Saud	Alganim	24 Alfadel Street	Dammam	27145	EE
20011212	Abdulrahman	Abdulsalam	10 Almadinah Street	Skaka	88756	CHEM
20011213	Salman	Alsaleh	15 King Fahad Road	Dammam	28898	PHYS
20011214	Khalid	Alomar	91 Alwadi Street	Najran	90987	MATH
20011215	Minwer	Almutairi	87 Alhamra Road	Jizan	92347	STAT
20011216	Turki	Alassaf	25 Prince Abdullah Street	Riyadh	11897	ENGL
20011217	Saleh	Alzaid	25 King Faisal Street	Riyadh	11874	EE
20021111	Ghanim	Alhmoud	56 Altahliah Street	Jeddah	35234	CS
20021212	Sultan	Abdulgader	123 Salman Alfarsi Street	Riyadh	12657	CHEM
20021213	Suliman	Almushari	45 Prince Sultan Street	Najran	90888	PHYS
20021214	Ahmad	Alsaif	13 Khalifa Street	Taif	67898	MATH
20021234	Ahmad	Alshemamri	15 Othman street	Jizan	92534	ENGL
20022345	Mohammed	Alzamil	67 Abubaker Road	Abha	56879	STAT
20023678	Mansour	Alzamil	13 King Abdulaziz Road	Tabouk	78453	EE

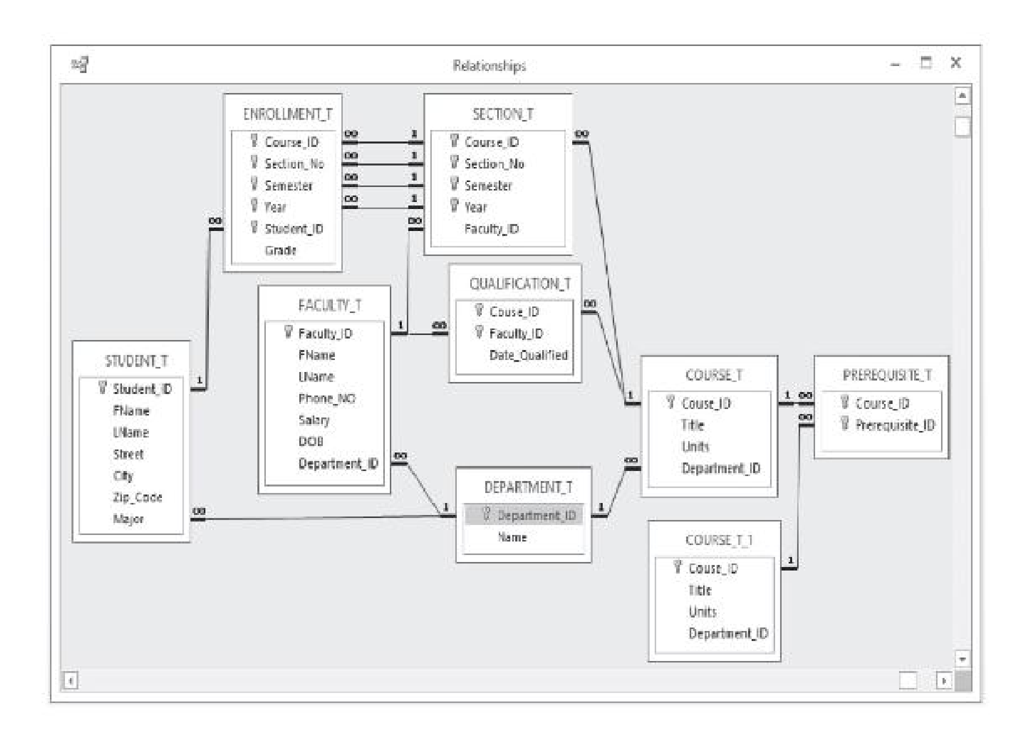
– جدول تسجيل الطلبة (ENROLLMENT_T):

- جدول تسجيل الطلبة (ENROLLMENT_T):

ENROLLMENT_T

Course ID	Section No	Semester	Year	Student ID	Grade
CHEM101	1	FALL	2000	19992020	4
CHEM101	1	FALL	2000	19992341	3
CHEM101	1	FALL	2000	20001212	4
CHEM101	2	FALL	2000	19994512	3
CHEM101	2	FALL	2000	20001111	1
CS101	1	FALL	2000	19992020	2
CS101	2	FALL	2000	20001111	4
CS102	1	SPRING	2000	19992020	3
CS102	1	SPRING	2000	20001111	4
ENGL101	1	FALL	2000	19992020	3
ENGL101	1	FALL	2000	19992341	4
ENGL101	1	FALL	2000	19994512	4
ENGL101	1	FALL	2000	20001111	4
ENGL102	1	SPRING	2000	19992020	1
ENGL102	1	SPRING	2000	20001111	4
MATH101	1	FALL	2000	19992020	3
MATH101	1	FALL	2000	19992341	2
MATH101	1	FALL	2000	19994512	0
MATH101	1	FALL	2000	20001111	2
MATH102	1	SPRING	2000	19992020	2
MATH102	1	SPRING	2000	20001111	0
STAT101	1	SPRING	2000	19992020	2
STAT101	1	SPRING	2000	20001111	3

ملحق رقم (1) – 5 العلاقات بين جداول قاعدة البيانات حسب بنائها باستخدام نظام إدارة قاعدة بيانات أكسس:



ملحق رقم (1) – 6 إنشاء قاعدة البيانات باستخدام تعليمات (SQL) في بيئة أوراكل (SQL*Plus):

عند إنشاء جداول قاعدة بيانات الجمعية الأهلية أو أية قاعدة بيانات أخرى، وعلى خلاف ما هو مُتَّبع في بيئة نظام قاعدة بيانات أكسس؛ يجب ملاحظة ترتيب إنشاء الجداول؛ بحيث يتمُّ إنشاء الجداول التي تحتوي على مفاتيح خارجية (لتمثيل العلاقات) بعد إنشاء الجداول التي تحتوي على المفاتيح الرئيسية التي تربطها بـالمفاتيح الخارجية. وفي حالة عدم اتباع ذلك؛ فإن نظام إدارة قاعدة البيانات لن يقوم بـإنشاء هذه الجداول وسيصدر عنه خطأ نتيجة لمثل هذا الإجراء. ويُعزَى السبب في ذلك، في هذه الحالة، إلى أن المُستخدِم يحاول إنشاء جدول يحتوي على حقل (أو حقول) ترتبط بحقول أخرى (وهي المفاتيح الرئيسية)، وهذه الحقول غير مُعرَّفة أصلاً في قاعدة البيانات. وفي حالة عدم رغبة المستخدِم في إنشاء الجداول، وفق ترتيب معين، أو في حالة وجود علاقات (أو قيود) متداخلة (CyCliC Relationships)؛ فإنه يمكِن إنشاء الجداول دون إنشاء العلاقات (أو القيود) فيما بينها. وبعد إنشاء الجداول تتمُّ إضافة العلاقات (أو القيود) باستخدام عبارة «تعديل جدول» (ALTER TABLE) (GarCia-Molina et al, 2014).

أمَّا في حالة نظام إدارة قاعدة بيانات أكسس؛ فإنه يتمُّ تمثيل العلاقات (أو القيود) ما بين جداول قاعدة البيانات في مرحلة تلي مرحلة إنشاء الجداول، ومن ثمَّ لا تظهر مثل هذه المشكلة للمستخدِم عند إنشائه للجداول والعلاقات فيما بينها.

– جدول الأقسام العلمية (DEPARTMENT_T):

```
CREATE TABLE DEPARTMENT_T
  (DEPARTMENT_ID          CHAR(6)       NOT NULL,
   NAME                   CHAR(30)      NOT NULL,
CONSTRAINT DEPARTMENT_PK PRIMARY KEY (DEPARTMENT_ID));
```

ويتمُّ إدخال البيانات للجدول باستخدام (SQL) حسب التعليمات التالية:

```
INSERT INTO DEPARTMENT_T VALUES ('CHEM', 'Chemistry');
INSERT INTO DEPARTMENT_T VALUES ('CS', 'Computer Science');
INSERT INTO DEPARTMENT_T VALUES ('EE', 'Electrical Engineering');
INSERT INTO DEPARTMENT_T VALUES ('ENGL', 'English Language');
INSERT INTO DEPARTMENT_T VALUES ('MATH', 'Mathematics');
INSERT INTO DEPARTMENT_T VALUES ('PHYS', 'Physics');
INSERT INTO DEPARTMENT_T VALUES ('STAT', 'Statistics');
```

- جدول المواد الدراسية (COURSE_T):

```
CREATE TABLE COURSE_T
        (COURSE_ID          CHAR(7)          NOT NULL,
         TITLE              CHAR(35)         NOT NULL,
         UNITS              NUMBER           NOT NULL,
         DEPARTMENT_ID      CHAR(6)          NOT NULL,
CONSTRAINT COURSE_PK PRIMARY KEY (COURSE_ID),
CONSTRAINT COURSE_FK1 FOREIGN KEY (DEPARTMENT_ID)
                REFERENCES DEPARTMENT_T(DEPARTMENT_ID));
```

ويتمُّ إدخال البيانات للجدول باستخدام (SQL) حسب التعليمات التالية:

```
INSERT INTO COURSE_T VALUES ('CHEM101', 'CHEMISTRY (I)', 3, 'CHEM');
INSERT INTO COURSE_T VALUES ('CHEM102', 'CHEMISTRY (II)', 3, 'CHEM');
INSERT INTO COURSE_T VALUES ('CS101', 'JAVA PROGRAMMING', 3, 'CS');
INSERT INTO COURSE_T VALUES ('CS102', 'SOFTWARE ENGINEERING', 3, 'CS');
INSERT INTO COURSE_T VALUES ('CS103', 'C/C++ PROGRAMMING', 3, 'CS');
INSERT INTO COURSE_T VALUES ('CS104', 'COMPUTER ARCHITECTURE', 3, 'CS');
INSERT INTO COURSE_T VALUES ('CS105', 'INTRODUCTION TO DATABASE SYSTEMS', 3, 'CS');
INSERT INTO COURSE_T VALUES ('EE101', 'ELECTRIC CIRCUITS', 3, 'EE');
INSERT INTO COURSE_T VALUES ('EE102', 'ELECTRONICS (I)', 3, 'EE');
INSERT INTO COURSE_T VALUES ('EE103', 'ELECTRONICS (II)', 3, 'EE');
INSERT INTO COURSE_T VALUES ('EE104', 'COMMUNICATION NETWORKS', 4, 'EE');
INSERT INTO COURSE_T VALUES ('ENGL101', 'ENGLISH GRAMMAR', 2, 'ENGL');
INSERT INTO COURSE_T VALUES ('ENGL102', 'ENGLISH WRITING', 3, 'ENGL');
INSERT INTO COURSE_T VALUES ('ENGL103', 'TECHNICAL WRITING', 3, 'ENGL');
INSERT INTO COURSE_T VALUES ('MATH101', 'INTRODUCTION TO MATHEMATICS', 3, 'MATH');
INSERT INTO COURSE_T VALUES ('MATH102', 'DIFFERENTIAL EQUATIONS', 3, 'MATH');
INSERT INTO COURSE_T VALUES ('MATH103', 'CALCULUS (I)', 3, 'MATH');
INSERT INTO COURSE_T VALUES ('MATH104', 'CALCULUS (II)', 3, 'MATH');
INSERT INTO COURSE_T VALUES ('MATH106', 'ALGEBRA', 4, 'MATH');
INSERT INTO COURSE_T VALUES ('MATH107', 'COMPUTER MATHEMATICS', 3, 'MATH');
INSERT INTO COURSE_T VALUES ('PHYS101', 'PHYSICS (I)', 3, 'PHYS');
INSERT INTO COURSE_T VALUES ('PHYS102', 'PHYSICS (II)', 3, 'PHYS');
INSERT INTO COURSE_T VALUES ('STAT101', 'INTRODUCTION TO STATISTICS', 3, 'STAT');
INSERT INTO COURSE_T VALUES ('STAT102', 'ADVANCED STATISTICS', 3, 'STAT');
```

- جدول المواد الدراسية المتطلبة (**PREREQUISITE_T**):

```
CREATE TABLE PREREQUISITE_T
        (COURSE_ID            CHAR(7)              NOT NULL,
         PREREQUISITE_ID      CHAR(7)              NOT NULL,
CONSTRAINT RREREQUISITE_PK PRIMARY KEY (COURSE_ID, PREREQUISITE_ID),
CONSTRAINT PREREQUISITE_FK1 FOREIGN KEY (COURSE_ID)
                               REFERENCES COURSE_T(COURSE_ID),
CONSTRAINT PREREQUISITE_FK2 FOREIGN KEY (PREREQUISITE_ID)
                               REFERENCES COURSE_T(COURSE_ID));
```

ويتمُّ إدخال البيانات للجدول باستخدام (SQL) حسب التعليمات التالية:

```
INSERT INTO PREREQUISITE_T VALUES ('CHEM102', 'CHEM101');
INSERT INTO PREREQUISITE_T VALUES ('CS102', 'MATH101');
INSERT INTO PREREQUISITE_T VALUES ('CS103', 'CS102');
INSERT INTO PREREQUISITE_T VALUES ('CS105', 'MATH101');
INSERT INTO PREREQUISITE_T VALUES ('EE102', 'EE101');
INSERT INTO PREREQUISITE_T VALUES ('EE103', 'EE102');
INSERT INTO PREREQUISITE_T VALUES ('EE103', 'MATH101');
INSERT INTO PREREQUISITE_T VALUES ('MATH102', 'MATH101');
INSERT INTO PREREQUISITE_T VALUES ('MATH103', 'MATH101');
INSERT INTO PREREQUISITE_T VALUES ('MATH104', 'MATH103');
INSERT INTO PREREQUISITE_T VALUES ('MATH106', 'MATH101');
INSERT INTO PREREQUISITE_T VALUES ('MATH107', 'MATH101');
INSERT INTO PREREQUISITE_T VALUES ('PHYS102', 'PHYS101');
INSERT INTO PREREQUISITE_T VALUES ('STAT102', 'STAT101');
```

- جدول أعضاء هيئة التدريس (**FUCULTY_T**):

```
CREATE TABLE FACULTY_T
        (FACULTY_ID           CHAR(8)              NOT NULL,
         FNAME                CHAR(12)             NOT NULL,
         LNAME                CHAR(12)             NOT NULL,
         PHONE_NO             CHAR(8),
         SALARY               NUMBER(9,2),
         DOB                  DATE                 NOT NULL,
         DEPARTMENT_ID        CHAR(6)              NOT NULL,
CONSTRAINT FACULTY_PK PRIMARY KEY (FACULTY_ID),
CONSTRAINT FACULTY_FK1 FOREIGN KEY (DEPARTMENT_ID)
                 REFERENCES DEPARTMENT_T(DEPARTMENT_ID));
```

ويتمُّ إدخال البيانات للجدول باستخدام (SQL) حسب التعليمات التالية:

```
INSERT INTO FACULTY_T
        VALUES ('200', 'Khalid', 'Aloufi', '454-2341', 35000, '22-05-1963', 'MATH');
INSERT INTO FACULTY_T
        VALUES ('220', 'Fahad', 'Alhamid', '456-7733', 25900, '07-10-1970', 'MATH');
INSERT INTO FACULTY_T
        VALUES ('310', 'Saleh', 'Aleesa', '454-8932', 30000, '13-09-1966', 'CS');
INSERT INTO FACULTY_T
        VALUES ('320', 'Mohammed', 'Alhamad', '454-5412', 44000, '13-05-1965', 'CS');
INSERT INTO FACULTY_T
        VALUES ('330', 'Ghanim', 'Alghanim', '456-2234', 44500, '12-08-1969', 'CS');
INSERT INTO FACULTY_T
        VALUES ('340', 'Ibraheem', 'Alsaleh', '454-1234', 25000, '20-01-1970', 'CS');
INSERT INTO FACULTY_T
        VALUES ('400', 'Ahmad', 'Alotaibi', '454-4563', 33900, '17-05-1971', 'CHEM');
INSERT INTO FACULTY_T
        VALUES ('420', 'Saleh', 'Alghamdi', '454-2233', 44600, '13-02-1969', 'CHEM');
INSERT INTO FACULTY_T
        VALUES ('500', 'Yahya', 'Khorshid', '456-2221', 36700, '12-03-1965', 'ENGL');
INSERT INTO FACULTY_T
        VALUES ('540', 'Salem', 'Alhamad', '456-3304', 40000, '11-09-1972', 'ENGL');
INSERT INTO FACULTY_T
        VALUES ('560', 'Salman', 'Albassam', '454-7865', 33800, '13-09-1968', 'ENGL');
INSERT INTO FACULTY_T
        VALUES ('600', 'Turki', 'Alturki', '456-7891', 27800, '23-07-1975', 'STAT');
INSERT INTO FACULTY_T
        VALUES ('640', 'Fahad', 'Alzaid', '456-3322', 44300, '12-05-1971', 'STAT');
INSERT INTO FACULTY_T
        VALUES ('660', 'Saud', 'Alkhalifa', '454-9856', 44900, '13-08-1972', 'STAT');
INSERT INTO FACULTY_T
        VALUES ('710', 'Mahmood', 'Alsalem', '456-3323', 31900, '19-02-1973', 'PHYS');
INSERT INTO FACULTY_T
        VALUES ('730', 'Mishal', 'Almazid', '454-2343', 29800, '17-09-1975', 'PHYS');
INSERT INTO FACULTY_T
        VALUES ('770', 'Sultan', 'Aljasir', '456-3212', 43300, '13-05-1970', 'PHYS');
INSERT INTO FACULTY_T
        VALUES ('800', 'Ali', 'Albader', '456-7812', 45300, '22-06-1966', 'EE');
INSERT INTO FACULTY_T
        VALUES ('810', 'Saad', 'Alzhrani', '454-5578', 44200, '17-10-1967', 'EE');
INSERT INTO FACULTY_T
        VALUES ('850', 'Ahmad', 'Alsabti', '456-0120', 33900, '15-04-1973', 'EE');
```

– جدول القدرات التعليمية لأعضاء هيئة التدريس (**QUALIFICATION_T**)

```
CREATE TABLE QUALIFICATION_T
        (COURSE_ID    CHAR(7)              NOT NULL,
         FACULTY_ID   CHAR(8)              NOT NULL,
         DATE_QUALIFIED DATE               NOT NULL,
CONSTRAINT QUALIFICATION_PK PRIMARY KEY (COURSE_ID, FACULTY_ID),
CONSTRAINT QUALIFICATION_FK1 FOREIGN KEY (COURSE_ID)
                    REFERENCES COURSE_T(COURSE_ID),
CONSTRAINT QUALIFICATION_FK2 FOREIGN KEY (FACULTY_ID)
                         REFERENCES FACULTY_T(FACULTY_ID));
```

ويتمُّ إدخال البيانات للجدول باستخدام (SQL) حسب التعليمات التالية:

```
INSERT INTO QUALIFICATION_T VALUES ('CHEM101', 400, '02-01-1991');
INSERT INTO QUALIFICATION_T VALUES ('CHEM102', 420, '02-07-1992');
INSERT INTO QUALIFICATION_T VALUES ('CS101', 310, '05-06-1995');
INSERT INTO QUALIFICATION_T VALUES ('CS102', 320, '09-08-1995');
INSERT INTO QUALIFICATION_T VALUES ('CS103', 320, '03-08-1996');
INSERT INTO QUALIFICATION_T VALUES ('CS104', 330, '02-09-1997');
INSERT INTO QUALIFICATION_T VALUES ('CS105', 340, '02-12-1997');
INSERT INTO QUALIFICATION_T VALUES ('EE101', 800, '08-01-1993');
INSERT INTO QUALIFICATION_T VALUES ('EE102', 810, '12-03-1994');
INSERT INTO QUALIFICATION_T VALUES ('EE103', 850, '15-11-1995');
INSERT INTO QUALIFICATION_T VALUES ('EE104', 810, '03-01-1996');
INSERT INTO QUALIFICATION_T VALUES ('ENGL101', 500, '01-07-1995');
INSERT INTO QUALIFICATION_T VALUES ('ENGL102', 540, '02-08-1994');
INSERT INTO QUALIFICATION_T VALUES ('ENGL103', 560, '09-09-1993');
INSERT INTO QUALIFICATION_T VALUES ('MATH101', 200, '13-11-1991');
INSERT INTO QUALIFICATION_T VALUES ('MATH102', 200, '02-06-1993');
INSERT INTO QUALIFICATION_T VALUES ('MATH103', 220, '02-07-1993');
INSERT INTO QUALIFICATION_T VALUES ('MATH104', 220, '13-08-1993');
INSERT INTO QUALIFICATION_T VALUES ('MATH106', 220, '17-10-1994');
INSERT INTO QUALIFICATION_T VALUES ('MATH107', 200, '10-01-1995');
INSERT INTO QUALIFICATION_T VALUES ('PHYS101', 710, '13-07-1996');
INSERT INTO QUALIFICATION_T VALUES ('PHYS101', 770, '11-02-1996');
INSERT INTO QUALIFICATION_T VALUES ('PHYS102', 730, '02-01-1997');
INSERT INTO QUALIFICATION_T VALUES ('STAT101', 600, '15-08-1993');
INSERT INTO QUALIFICATION_T VALUES ('STAT101', 660, '03-04-1995');
INSERT INTO QUALIFICATION_T VALUES ('STAT102', 640, '02-05-1994');
```

– **جدول المجموعات (أو الشعب) الدراسية (SECTION_T):**

```
CREATE TABLE SECTION_T
        (COURSE_ID    CHAR(7)           NOT NULL,
         SECTION_NO   NUMBER            NOT NULL,
         SEMESTER     CHAR(10)          NOT NULL,
         YEAR         NUMBER            NOT NULL,
         FACULTY_ID   CHAR(8)           NOT NULL,
         LOCATION     CHAR(12)          NOT NULL,
CONSTRAINT SECTION_PK PRIMARY KEY
                          (COURSE_ID, SECTION_NO, SEMESTER, YEAR),
CONSTRAINT SECTION_FK1 FOREIGN KEY (COURSE_ID)
                          REFERENCES COURSE_T(COURSE_ID),
CONSTRAINT SECTION_FK2 FOREIGN KEY (FACULTY_ID)
                                REFERENCES FACULTY_T(FACULTY_ID));
```

ويتمُّ إدخال البيانات للجدول باستخدام (SQL) حسب التعليمات التالية:

```
INSERT INTO SECTION_T VALUES ('CHEM101', 1, 'FALL', 2000, 400, 'Riyadh');
INSERT INTO SECTION_T VALUES ('CHEM101', 2, 'FALL', 2000, 400, 'Dammam');
INSERT INTO SECTION_T VALUES ('CS101', 1, 'FALL', 2000, 310, 'Jeddah');
INSERT INTO SECTION_T VALUES ('CS101', 2, 'FALL', 2000, 310, 'Abha');
INSERT INTO SECTION_T VALUES ('CS102', 1, 'SPRING', 2000, 320, 'Riyadh');
INSERT INTO SECTION_T VALUES ('CS103', 1, 'SPRING', 2000, 320, 'Riyadh');
INSERT INTO SECTION_T VALUES ('CS104', 1, 'FALL', 2001, 330, 'Riyadh');
INSERT INTO SECTION_T VALUES ('CS105', 1, 'SPRING', 2001, 340, 'Dammam');
INSERT INTO SECTION_T VALUES ('EE101', 1, 'FALL', 2001, 800, 'Abha');
INSERT INTO SECTION_T VALUES ('EE102', 1, 'SPRING', 2001, 810, 'Jeddah');
INSERT INTO SECTION_T VALUES ('ENGL101', 1, 'FALL', 2000, 500, 'Dammam');
INSERT INTO SECTION_T VALUES ('ENGL102', 1, 'SPRING', 2000, 540, 'Jeddah');
INSERT INTO SECTION_T VALUES ('MATH101', 1, 'FALL', 2000, 200, 'Riyadh');
INSERT INTO SECTION_T VALUES ('MATH102', 1, 'SPRING', 2000, 200, 'Jeddah');
INSERT INTO SECTION_T VALUES ('MATH103', 1, 'FALL', 2001, 220, 'Abha');
INSERT INTO SECTION_T VALUES ('MATH104', 1, 'SPRING', 2001, 220, 'Jeddah');
INSERT INTO SECTION_T VALUES ('PHYS101', 1, 'FALL', 2001, 710, 'Abha');
INSERT INTO SECTION_T VALUES ('PHYS102', 1, 'SPRING', 2001, 730, 'Riyadh');
INSERT INTO SECTION_T VALUES ('STAT101', 1, 'SPRING', 2000, 600, 'Abha');
INSERT INTO SECTION_T VALUES ('STAT102', 1, 'SPRING', 2001, 640, 'Jeddah');
```

– جدول الطلاب (STUDENT_T):

```
CREATE TABLE STUDENT_T
        (STUDENT_ID        CHAR(8)          NOT NULL,
         FNAME             CHAR(12)         NOT NULL,
         LNAME             CHAR(12)         NOT NULL,
         STREET            CHAR(30),
         CITY              CHAR(8),
         ZIP_CODE          CHAR(5),
         MAJOR             CHAR(6)          NOT NULL,
CONSTRAINT STUDENT_PK  PRIMARY KEY (STUDENT_ID),
CONSTRAINT STUDENT_FK1 FOREIGN KEY (MAJOR)
              REFERENCES DEPARTMENT_T(DEPARTMENT_ID));
```

ويتمُّ إدخال البيانات للجدول باستخدام (SQL) حسب التعليمات التالية:

```
INSERT INTO STUDENT_T
VALUES ('19992020', 'Saleh', 'Alhamad', '13 Almutanabi Street', 'Riyadh', '11121', 'CS');
INSERT INTO STUDENT_T
VALUES ('19992341', 'Abdullah', 'Aloufi', '25 Jareer Street', 'Riyadh', '12123', 'CHEM');
INSERT INTO STUDENT_T
VALUES ('19994512', 'Salem', 'Algamdi', '98 Bin Taimiah Street', 'Jeddah', '34565', 'PHYS');
INSERT INTO STUDENT_T
VALUES ('20001111', 'Mishal', 'Alyousef', '13 Alsouk Street', 'Taif', '67156', 'CS');
INSERT INTO STUDENT_T
VALUES ('20001212', 'Khalid', 'Alsultan', '22 Bin Hamdan Street', 'Jeddah', '34565', 'MATH');
INSERT INTO STUDENT_T
VALUES ('20001213', 'Mohammed', 'Abdelaleem', '10 Bin Hamdan Street', 'Jeddah', '35787', 'STAT');
INSERT INTO STUDENT_T
VALUES ('20001214', 'Sami', 'Aloutaibi', '67 Alfadel Street', 'Dammam', '26123', 'ENGL');
INSERT INTO STUDENT_T
VALUES ('20001215', 'Saud', 'Alganim', '24 Alfadel Street', 'Dammam', '27145', 'EE');
INSERT INTO STUDENT_T
VALUES ('20011212', 'Abdulrahman', 'Abdulsalam', '10 Almadinah Street', 'Skaka', '88756', 'CHEM');
INSERT INTO STUDENT_T
VALUES ('20011213', 'Salman', 'Alsaleh', '15 King Fahad Road', 'Dammam', '28898', 'PHYS');
INSERT INTO STUDENT_T
VALUES ('20011214', 'Khalid', 'Alomar', '91 Alwadi Street', 'Najran', '90987', 'MATH');
INSERT INTO STUDENT_T
VALUES ('20011215', 'Minwer', 'Almutairi', '87 Alhamra Road', 'Jizan', '92347', 'STAT');
INSERT INTO STUDENT_T
VALUES ('20011216', 'Turki', 'Alassaf', '25 Prince Abdullah Street', 'Riyadh', '11897', 'ENGL');
INSERT INTO STUDENT_T
VALUES ('20011217', 'Saleh', 'Alzaid', '25 King Faisal Street', 'Riyadh', '11874', 'EE');
INSERT INTO STUDENT_T
VALUES ('20021111', 'Ghanim', 'Alhmoud', '56 Altahliah Street', 'Jeddah', '35234', 'CS');
INSERT INTO STUDENT_T
VALUES ('20021212', 'Sultan', 'Abdulgader', '123 Salman Alfarsi Street', 'Riyadh', '12657', 'CHEM');
INSERT INTO STUDENT_T
VALUES ('20021213', 'Suliman', 'Almushari', '45 Prince Sultan Street', 'Najran', '90888', 'PHYS');
INSERT INTO STUDENT_T
VALUES ('20021214', 'Ahmad', 'Alsaif', '13 Khalifa Street', 'Taif', '67898', 'MATH');
INSERT INTO STUDENT_T
VALUES ('20021234', 'Ahmad', 'Alshemamri', '15 Othman street', 'Jizan', '92534', 'ENGL');
INSERT INTO STUDENT_T
VALUES ('20022345', 'Mohammed', 'Alzamil', '67 Abubaker Road', 'Abha', '56879', 'STAT');
INSERT INTO STUDENT_T
VALUES ('20023678', 'Mansour', 'Alzamil', '13 King Abdulaziz Road', 'Tabouk', '78453', 'EE');
```

- **جدول تسجيل الطلبة (ENROLLMENT_T):**

```
CREATE TABLE ENROLLMENT_T
        (COURSE_ID    CHAR(7)          NOT NULL,
         SECTION_NO   NUMBER           NOT NULL,
         SEMESTER     CHAR(10)         NOT NULL,
         YEAR         NUMBER           NOT NULL,
         STUDENT_ID   CHAR(8)          NOT NULL,
         GRADE        NUMBER,
CONSTRAINT ENROLLMENT_PK PRIMARY KEY
        (COURSE_ID, SECTION_NO, SEMESTER, YEAR, STUDENT_ID),
CONSTRAINT ENROLLMENT_FK1 FOREIGN KEY
        (COURSE_ID, SECTION_NO, SEMESTER, YEAR)
        REFERENCES SECTION_T(COURSE_ID, SECTION_NO, SEMESTER, YEAR),
CONSTRAINT ENROLLMENT_FK2 FOREIGN KEY (STUDENT_ID)
        REFERENCES STUDENT_T(STUDENT_ID));
```

ويتمُّ إدخال البيانات للجدول باستخدام (SQL) حسب التعليمات التالية:

```
INSERT INTO ENROLLMENT_T VALUES ('CHEM101', 1, 'FALL', 2000, '19992020', 4);
INSERT INTO ENROLLMENT_T VALUES ('CHEM101', 1, 'FALL', 2000, '19992341', 3);
INSERT INTO ENROLLMENT_T VALUES ('CHEM101', 1, 'FALL', 2000, '20001212', 4);
INSERT INTO ENROLLMENT_T VALUES ('CHEM101', 2, 'FALL', 2000, '19994512', 3);
INSERT INTO ENROLLMENT_T VALUES ('CHEM101', 2, 'FALL', 2000, '20001111', 1);
INSERT INTO ENROLLMENT_T VALUES ('CS101', 1, 'FALL', 2000, '19992020', 2);
INSERT INTO ENROLLMENT_T VALUES ('CS101', 2, 'FALL', 2000, '20001111', 4);
INSERT INTO ENROLLMENT_T VALUES ('CS102', 1, 'SPRING', 2000, '19992020', 3);
INSERT INTO ENROLLMENT_T VALUES ('CS102', 1, 'SPRING', 2000, '20001111', 4);
INSERT INTO ENROLLMENT_T VALUES ('ENGL101', 1, 'FALL', 2000, '19992020', 3);
INSERT INTO ENROLLMENT_T VALUES ('ENGL101', 1, 'FALL', 2000, '19992341', 4);
INSERT INTO ENROLLMENT_T VALUES ('ENGL101', 1, 'FALL', 2000, '19994512', 4);
INSERT INTO ENROLLMENT_T VALUES ('ENGL101', 1, 'FALL', 2000, '20001111', 4);
INSERT INTO ENROLLMENT_T VALUES ('ENGL102', 1, 'SPRING', 2000, '19992020', 1);
INSERT INTO ENROLLMENT_T VALUES ('ENGL102', 1, 'SPRING', 2000, '20001111', 4);
INSERT INTO ENROLLMENT_T VALUES ('MATH101', 1, 'FALL', 2000, '19992020', 3);
INSERT INTO ENROLLMENT_T VALUES ('MATH101', 1, 'FALL', 2000, '19992341', 2);
INSERT INTO ENROLLMENT_T VALUES ('MATH101', 1, 'FALL', 2000, '19994512', 0);
INSERT INTO ENROLLMENT_T VALUES ('MATH101', 1, 'FALL', 2000, '20001111', 2);
INSERT INTO ENROLLMENT_T VALUES ('MATH102', 1, 'SPRING', 2000, '19992020', 2);
INSERT INTO ENROLLMENT_T VALUES ('MATH102', 1, 'SPRING', 2000, '20001111', 0);
INSERT INTO ENROLLMENT_T VALUES ('STAT101', 1, 'SPRING', 2000, '19992020', 2);
INSERT INTO ENROLLMENT_T VALUES ('STAT101', 1, 'SPRING', 2000, '20001111', 3);
```

ملحق رقم (1) – 7 استعراض لمحتويات جداول قاعدة البيانات بعد إنشائها في بيئة أوراكل باستخدام تعليمة (**SELECT * FROM TableName**):

- جدول الأقسام العلمية (**DEPARTMENT_T**):

DEPART	NAME
CHEM	Chemistry
CS	Computer Science
EE	Electrical Engineering
ENGL	English Language
MATH	Mathematics
PHYS	Physics
STAT	Statistics

- جدول المواد الدراسية (**COURSE_T**):

COURSE_	TITLE	UNITS	DEPART
CHEM101	CHEMISTRY (I)	3	CHEM
CHEM102	CHEMISTRY (II)	3	CHEM
CS101	JAVA PROGRAMMING	3	CS
CS102	SOFTWARE ENGINEERING	3	CS
CS103	C/C++ PROGRAMMING	3	CS
CS104	COMPUTER ARCHITECTURE	3	CS
CS105	INTRODUCTION TO DATABASE SYSTEMS	3	CS
EE101	ELECTRIC CIRCUITS	3	EE
EE102	ELECTRONICS (I)	3	EE
EE103	ELECTRONICS (II)	3	EE
EE104	COMMUNICATION NETWORKS	4	EE
ENGL101	ENGLISH GRAMMAR	2	ENGL
ENGL102	ENGLISH WRITING	3	ENGL
ENGL103	TECHNICAL WRITING	3	ENGL
MATH101	INTRODUCTION TO MATHEMATICS	3	MATH
MATH102	DIFFERENTIAL EQUATIONS	3	MATH
MATH103	CALCULUS (I)	3	MATH
MATH104	CALCULUS (II)	3	MATH
MATH106	ALGEBRA	4	MATH
MATH107	COMPUTER MATHEMATICS	3	MATH
PHYS101	PHYSICS (I)	3	PHYS
PHYS102	PHYSICS (II)	3	PHYS
STAT101	INTRODUCTION TO STATISTICS	3	STAT
STAT102	ADVANCED STATISTICS	3	STAT

- جدول المواد الدراسية المتطلبة (**PREREQUISITE_T**):

COURSE_	PREREQU
CHEM102	CHEM101
CS102	MATH101
CS103	CS102
CS105	MATH101
EE102	EE101
EE103	EE102
EE103	MATH101
MATH102	MATH101
MATH103	MATH101
MATH104	MATH103
MATH106	MATH101
MATH107	MATH101
PHYS102	PHYS101
STAT102	STAT101

- جدول أعضاء هيئة التدريس (**FUCULTY_T**):

FACULTY_	FNAME	LNAME	PHONE_NO	SALARY	DOB	DEPART
200	Khalid	Aloufi	454-2341	35000	22/05/63	MATH
220	Fahad	Alhamid	456-7733	25900	07/10/70	MATH
310	Saleh	Aleesa	454-8932	30000	13/09/66	CS
320	Mohammed	Alhamad	454-5412	44000	13/05/65	CS
330	Ghanim	Alghanim	456-2234	44500	12/08/69	CS
340	Ibraheem	Alsaleh	454-1234	25000	20/01/70	CS
400	Ahmad	Alotaibi	454-4563	33900	17/05/71	CHEM
420	Saleh	Alghamdi	454-2233	44600	13/02/69	CHEM
500	Yahya	Khorshid	456-2221	36700	12/03/65	ENGL
540	Salem	Alhamad	456-3304	40000	11/09/72	ENGL
560	Salman	Albassam	454-7865	33800	13/09/68	ENGL
600	Turki	Alturki	456-7891	27800	23/07/75	STAT
640	Fahad	Alzaid	456-3322	44300	12/05/71	STAT
660	Saud	Alkhalifa	454-9856	44900	13/08/72	STAT
710	Mahmood	Alsalem	456-3323	31900	19/02/73	PHYS
730	Mishal	Almazid	454-2343	29800	17/09/75	PHYS
770	Sultan	Aljasir	456-3212	43300	13/05/70	PHYS
800	Ali	Albader	456-7812	45300	22/06/66	EE
810	Saad	Alzhrani	454-5578	44200	17/10/67	EE
850	Ahmad	Alsabti	456-0120	33900	15/04/73	EE

– جدول القدرات التعليمية لأعضاء هيئة التدريس (QUALIFICATION_T):

COURSE_	FACULTY_	DATE_QUA
CHEM101	400	02/01/91
CHEM102	420	02/07/92
CS101	310	05/06/95
CS102	320	09/08/95
CS103	320	03/08/96
CS104	330	02/09/97
CS105	340	02/12/97
EE101	800	08/01/93
EE102	810	12/03/94
EE103	850	15/11/95
EE104	810	03/01/96
ENGL101	500	01/07/95
ENGL102	540	02/08/94
ENGL103	560	09/09/93
MATH101	200	13/11/91
MATH102	200	02/06/93
MATH103	220	02/07/93
MATH104	220	13/08/93
MATH106	220	17/10/94
MATH107	200	10/01/95
PHYS101	710	13/07/96
PHYS101	770	11/02/96
PHYS102	730	02/01/97
STAT101	600	15/08/93
STAT101	660	03/04/95
STAT102	640	02/05/94

– جدول المجموعات (أو الشُّعَب) الدراسية (SECTION_T):

COURSE_	SECTION_NO	SEMESTER	YEAR	FACULTY_	LOCATION
CHEM101	1	FALL	2000	400	Riyadh
CHEM101	2	FALL	2000	400	Dammam
CS101	1	FALL	2000	310	Jeddah
CS101	2	FALL	2000	310	Abha
CS102	1	SPRING	2000	320	Riyadh
CS103	1	SPRING	2000	320	Riyadh
CS104	1	FALL	2001	330	Riyadh
CS105	1	SPRING	2001	340	Dammam
EE101	1	FALL	2001	800	Abha
EE102	1	SPRING	2001	810	Jeddah
ENGL101	1	FALL	2000	500	Dammam
ENGL102	1	SPRING	2000	540	Jeddah
MATH101	1	FALL	2000	200	Riyadh
MATH102	1	SPRING	2000	200	Jeddah
MATH103	1	FALL	2001	220	Abha
MATH104	1	SPRING	2001	220	Jeddah
PHYS101	1	FALL	2001	710	Abha
PHYS102	1	SPRING	2001	730	Riyadh
STAT101	1	SPRING	2000	600	Abha
STAT102	1	SPRING	2001	640	Jeddah

- جدول الطلاب (**STUDENT_T**):

STUDENT_	FNAME	LNAME	STREET	CITY	ZIP_C	MAJOR
19992020	Saleh	Alhamad	13 Almutanabi Street	Riyadh	11121	CS
19992341	Abdullah	Aloufi	25 Jareer Street	Riyadh	12123	CHEM
19994512	Salem	Algamdi	98 Bin Taimiah Street	Jeddah	34565	PHYS
20001111	Mishal	Alyousef	13 Alsouk Street	Taif	67156	CS
20001212	Khalid	Alsultan	22 Bin Hamdan Street	Jeddah	34565	MATH
20001213	Mohammed	Abdelaleem	10 Bin Hamdan Street	Jeddah	35787	STAT
20001214	Sami	Aloutaibi	67 Alfadel Street	Damman	26123	ENGL
20001215	Saud	Alganim	24 Alfadel Street	Damman	27145	EE
20011212	Abdulrahman	Abdulsalam	10 Almadinah Street	Skaka	88756	CHEM
20011213	Salman	Alsaleh	15 King Fahad Road	Damman	28898	PHYS
20011214	Khalid	Alomar	91 Alwadi Street	Najran	90987	MATH
20011215	Minwer	Almutairi	87 Alhamra Road	Jizan	92347	STAT
20011216	Turki	Alassaf	25 Prince Abdullah Street	Riyadh	11897	ENGL
20011217	Saleh	Alzaid	25 King Faisal Street	Riyadh	11874	EE
20021111	Ghanim	Alhnoud	56 Altahliah Street	Jeddah	35234	CS
20021212	Sultan	Abdulgader	123 Salman Alfarsi Street	Riyadh	12657	CHEM
20021213	Suliman	Almushari	45 Prince Sultan Street	Najran	90888	PHYS
20021214	Ahmad	Alsaif	13 Khalifa Street	Taif	67898	MATH
20021234	Ahmad	Alshenamri	15 Othman street	Jizan	92534	ENGL
20022345	Mohammed	Alzamil	67 Abubaker Road	Abha	56879	STAT
20023678	Mansour	Alzamil	13 King Abdulaziz Road	Tabouk	78453	EE

- جدول تسجيل الطلبة (**ENROLLMENT_T**):

COURSE_	SECTION_NO	SEMESTER	YEAR	STUDENT_	GRADE
CHEM101	1	FALL	2000	19992020	4
CHEM101	1	FALL	2000	19992341	3
CHEM101	1	FALL	2000	20001212	4
CHEM101	2	FALL	2000	19994512	3
CHEM101	2	FALL	2000	20001111	1
CS101	1	FALL	2000	19992020	2
CS101	2	FALL	2000	20001111	4
CS102	1	SPRING	2000	19992020	3
CS102	1	SPRING	2000	20001111	4
ENGL101	1	FALL	2000	19992020	3
ENGL101	1	FALL	2000	19992341	4
ENGL101	1	FALL	2000	19994512	4
ENGL101	1	FALL	2000	20001111	4
ENGL102	1	SPRING	2000	19992020	1
ENGL102	1	SPRING	2000	20001111	4
MATH101	1	FALL	2000	19992020	3
MATH101	1	FALL	2000	19992341	2
MATH101	1	FALL	2000	19994512	0
MATH101	1	FALL	2000	20001111	2
MATH102	1	SPRING	2000	19992020	2
MATH102	1	SPRING	2000	20001111	0
STAT101	1	SPRING	2000	19992020	2
STAT101	1	SPRING	2000	20001111	3

ملحق رقم (1) – 8 تمارين تطبيقية على لغة الاستفسار البنائية (SQL):

1– ما أرقام أعضاء هيئة التدريس العاملين في الجامعة؟

الحل:

```
SELECT FaCulty_ID
FROM FACULTY_T;
```

النتيجة:

```
FACULTY_
--------
200
220
310
320
330
340
400
420
500
540
560
600
640
660
710
730
770
800
810
850
```

2– ما هي كلُّ بيانات أعضاء هيئة التدريس العاملين في الجامعة؟

الحل:

```
SELECT *
FROM
FACULTY_T;
```

النتيجة:

FACULTY_	FNAME	LNAME	PHONE_NO	SALARY	DOB	DEPART
200	Khalid	Aloufi	454-2341	35000	22/05/63	MATH
220	Fahad	Alhamid	456-7733	25900	07/10/70	MATH
310	Saleh	Aleesa	454-8932	30000	13/09/66	CS
320	Mohammed	Alhamad	454-5412	44000	13/05/65	CS
330	Ghanim	Alghanim	456-2234	44500	12/08/69	CS
340	Ibraheem	Alsaleh	454-1234	25000	20/01/70	CS
400	Ahmad	Alotaibi	454-4563	33900	17/05/71	CHEM
420	Saleh	Alghamdi	454-2233	44600	13/02/69	CHEM
500	Yahya	Khorshid	456-2221	36700	12/03/65	ENGL
540	Salem	Alhamad	456-3304	40000	11/09/72	ENGL
560	Salman	Albassam	454-7865	33800	13/09/68	ENGL
600	Turki	Alturki	456-7891	27800	23/07/75	STAT
640	Fahad	Alzaid	456-3322	44300	12/05/71	STAT
660	Saud	Alkhalifa	454-9856	44900	13/08/72	STAT
710	Mahmood	Alsalem	456-3323	31900	19/02/73	PHYS
730	Mishal	Almazid	454-2343	29800	17/09/75	PHYS
770	Sultan	Aljasir	456-3212	43300	13/05/70	PHYS
800	Ali	Albader	456-7812	45300	22/06/66	EE
810	Saad	Alzhrani	454-5578	44200	17/10/67	EE
850	Ahmad	Alsabti	456-0120	33900	15/04/73	EE

3- ما أرقام المواد الدراسية وأرقام المواد الدراسية المتطلبة لكلٍّ منها؟

الحل:

```
SELECT Course_ID,
Prerequisite_ID
;FROM PREREQUISITE_T
```

النتيجة:

COURSE_	PREREQU
CHEM102	CHEM101
CS102	MATH101
CS103	CS102
CS105	MATH101
EE102	EE101
EE103	EE102
EE103	MATH101
MATH102	MATH101
MATH103	MATH101
MATH104	MATH103
MATH106	MATH101
MATH107	MATH101
PHYS102	PHYS101
STAT102	STAT101

4- ما أرقام أعضاء هيئة التدريس وأرقام البرامج المؤهلين لتدريسها؟

الحل:

```
SELECT FaCulty_ID,
Course_ID
FROM QUALIFICATION_T;
```

النتيجة:

FACULTY_	COURSE_
400	CHEM101
420	CHEM102
310	CS101
320	CS102
320	CS103
330	CS104
340	CS105
800	EE101
810	EE102
850	EE103
810	EE104
500	ENGL101
540	ENGL102
560	ENGL103
200	MATH101
200	MATH102
220	MATH103
220	MATH104
220	MATH106
200	MATH107
710	PHYS101
770	PHYS101
730	PHYS102
600	STAT101
660	STAT101
640	STAT102

5- ما بيانات أعضاء هيئة التدريس الذين يعملون في قسم الحاسب الآلي (CS)؟

الحل:

```
*SELECT
FROM FACULTY_T
WHERE Department_ID =
;'CS'
```

النتيجة:

FACULTY_	FNAME	LNAME	PHONE_NO	SALARY	DOB	DEPART
310	Saleh	Aleesa	454-8932	30000	13/09/66	CS
320	Mohammed	Alhamad	454-5412	44000	13/05/65	CS
330	Ghanim	Alghanim	456-2234	44500	12/08/69	CS
340	Ibraheem	Alsaleh	454-1234	25000	20/01/70	CS

6- ما أرقام وأسماء وأرقام هواتف أعضاء هيئة التدريس الذين يعملون في قسم اللغة الإنجليزية (ENGL)؟

الحل:

```
SELECT FaCulty_ID, FName, LName,
Phone_No
FROM FACULTY_T
WHERE Department_ID = ‘ENGL’;
```

النتيجة:

FACULTY_	FNAME	LNAME	PHONE_NO
500	Yahya	Khorshid	456-2221
540	Salem	Alhamad	456-3304
560	Salman	Albassam	454-7865

7- ما أسماء الطلبة من مدينة الرياض (Riyadh)؟

الحل:

```
SELECT FName,
LName
FROM STUDENT_T
WHERE CITY =
‘Riyadh’;
```

النتيجة:

```
FNAME        LNAME
------------ ------------
Saleh        Alhamad
Abdullah     Aloufi
Turki        Alassaf
Saleh        Alzaid
Sultan       Abdulgader
```

8- ما أرقام وأسماء وتواريخ ميلاد (DOB) أعضاء هيئة التدريس الذين تواريخ ميلادهم أكبر من ('20-01-1970')؟

الحل:

```
SELECT FaCulty_ID, FName,
LName, DOB
FROM FACULTY_T
;'WHERE DOB > '20-01-1970
```

النتيجة:

```
FACULTY_ FNAME        LNAME        DOB
-------- ------------ ------------ --------
220      Fahad        Alhamid      07/10/70
400      Ahmad        Alotaibi     17/05/71
540      Salem        Alhamad      11/09/72
600      Turki        Alturki      23/07/75
640      Fahad        Alzaid       12/05/71
660      Saud         Alkhalifa    13/08/72
710      Mahmood      Alsalem      19/02/73
730      Mishal       Almazid      17/09/75
770      Sultan       Aljasir      13/05/70
850      Ahmad        Alsabti      15/04/73
```

9- ما أرقام وأسماء أعضاء هيئة التدريس المؤهلين لتدريس المادة الدراسية (STAT101)؟

الحل:

```
SELECT FACULTY_T.FaCulty_ID, FName, LName
FROM FACULTY_T, QUALIFICATION_T
WHERE FACULTY_T.FaCulty_ID =
QUALIFICATION_T.FaCulty_ID
```

;’STAT101‘ AND Course_ID =

النتيجة:

```
FACULTY_ FNAME        LNAME
-------- ------------ ------------
600      Turki        Alturki
660      Saud         Alkhalifa
```

10- ما أرقام وأسماء وتقديرات الطلبة الذين درسوا المادة الدراسية (STAT101)؟

الحل:

```
SELECT STUDENT_T.Student_ID, FName, LName, Grade
FROM STUDENT_T, ENROLLMENT_T
WHERE STUDENT_T. Student_ID = ENROLLMENT_T.Student_ID
;'STAT101'AND Course_ID =
```

النتيجة:

```
STUDENT_ FNAME       LNAME            GRADE
-------- ----------- ----------- ----------
19992020 Saleh       Alhamad              2
20001111 Mishal      Alyousef             3
```

11- ما أرقام المواد الدراسية التي لم تنفذ في فصل الخريف (FALL) من عام (2000)؟

الحل:

```
SELECT Course_ID
FROM COURSE_T
WHERE Course_ID NOT IN
(SELECT Course_ID
```

```
FROM SECTION_T
WHERE Semester = 'FALL' AND Year =
;(2000
```

النتيجة:

```
COURSE_
-------
CHEM102
CS102
CS103
CS104
CS105
EE101
EE102
EE103
EE104
ENGL102
ENGL103
MATH102
MATH103
MATH104
MATH106
MATH107
PHYS101
PHYS102
STAT101
STAT102
```

12- ما أسماء أعضاء هيئة التدريس وأسماء الأقسام التي يتبعونها مرتبةً تصاعدياً حسب اختصارات أسماء الأقسام التي يتبعونها؟

الحل:

```
SELECT FName, LName, Name
FROM FACULTY_T, DEPARTMENT_T
```

```
WHERE FACULTY_T.Department_ID =
DEPARTMENT_T.Department_ID
ORDER BY FACULTY_T.Department_ID;
```

النتيجة:

```
FNAME        LNAME        NAME
------------ ------------ ------------------------------
Ahmad        Alotaibi     Chemistry
Saleh        Alghamdi     Chemistry
Mohammed     Alhamad      Computer Science
Saleh        Aleesa       Computer Science
Ghanim       Alghanim     Computer Science
Ibraheem     Alsaleh      Computer Science
Ahmad        Alsabti      Electrical Engineering
Ali          Albader      Electrical Engineering
Saad         Alzhrani     Electrical Engineering
Salman       Albassam     English Language
Yahya        Khorshid     English Language
Salem        Alhamad      English Language
Fahad        Alhamid      Mathematics
Khalid       Aloufi       Mathematics
Mahmood      Alsalem      Physics
Mishal       Almazid      Physics
Sultan       Aljasir      Physics
Fahad        Alzaid       Statistics
Saud         Alkhalifa    Statistics
Turki        Alturki      Statistics
```

13- ما أسماء أعضاء هيئة التدريس وتواريخ ميلادهم وأسماء الأقسام التي يتبعونها مرتبةً تصاعدياً حسب تواريخ ميلادهم وتصاعدياً داخل أسماء الأقسام التي يتبعونها (كاملة)؟

الحل:

```
SELECT FName, LName, DOB, Name
FROM FACULTY_T, DEPARTMENT_T
WHERE FACULTY_T.Department_ID =
DEPARTMENT_T.Department_ID
```

```
;ORDER BY Name, DOB
```

النتيجة:

```
FNAME        LNAME        DOB      NAME
------------ ------------ -------- ------------------------------
Saleh        Alghamdi     13/02/69 Chemistry
Ahmad        Alotaibi     17/05/71 Chemistry
Mohammed     Alhamad      13/05/65 Computer Science
Saleh        Aleesa       13/09/66 Computer Science
Ghanim       Alghanim     12/08/69 Computer Science
Ibraheem     Alsaleh      20/01/70 Computer Science
Ali          Albader      22/06/66 Electrical Engineering
Saad         Alzhrani     17/10/67 Electrical Engineering
Ahmad        Alsabti      15/04/73 Electrical Engineering
Yahya        Khorshid     12/03/65 English Language
Salman       Albassam     13/09/68 English Language
Salem        Alhamad      11/09/72 English Language
Khalid       Aloufi       22/05/63 Mathematics
Fahad        Alhamid      07/10/70 Mathematics
Sultan       Aljasir      13/05/70 Physics
Mahmood      Alsalem      19/02/73 Physics
Mishal       Almazid      17/09/75 Physics
Fahad        Alzaid       12/05/71 Statistics
Saud         Alkhalifa    13/08/72 Statistics
Turki        Alturki      23/07/75 Statistics
```

14- ما أسماء أعضاء هيئة التدريس وتواريخ ميلادهم وأسماء الأقسام التي يتبعونها مرتبة تنازلياً حسب تواريخ ميلادهم وتنازلياً داخل أسماء الأقسام التي يتبعونها (كاملة)؟

الحل:

```
SELECT FName, LName, DOB, Name
FROM FACULTY_T, DEPARTMENT_T
WHERE FACULTY_T.Department_ID =
DEPARTMENT_T.Department_ID
ORDER BY Name DESC, DOB DESC;
```

النتيجة:

```
FNAME        LNAME        DOB      NAME
------------ ------------ -------- ----------------------------
Turki        Alturki      23/07/75 Statistics
Saud         Alkhalifa    13/08/72 Statistics
Fahad        Alzaid       12/05/71 Statistics
Mishal       Almazid      17/09/75 Physics
Mahmood      Alsalem      19/02/73 Physics
Sultan       Aljasir      13/05/70 Physics
Fahad        Alhamid      07/10/70 Mathematics
Khalid       Aloufi       22/05/63 Mathematics
Salem        Alhamad      11/09/72 English Language
Salman       Albassam     13/09/68 English Language
Yahya        Khorshid     12/03/65 English Language
Ahmad        Alsabti      15/04/73 Electrical Engineering
Saad         Alzhrani     17/10/67 Electrical Engineering
Ali          Albader      22/06/66 Electrical Engineering
Ibraheem     Alsaleh      20/01/70 Computer Science
Ghanim       Alghanim     12/08/69 Computer Science
Saleh        Aleesa       13/09/66 Computer Science
Mohammed     Alhamad      13/05/65 Computer Science
Ahmad        Alotaibi     17/05/71 Chemistry
Saleh        Alghamdi     13/02/69 Chemistry
```

15- ما أرقام الطلبة ومعدلاتهم التراكمية؟

الحل:

```
SELECT S.Student_ID, SUM(Grade * Units) / SUM(Units) GPA
FROM STUDENT_T S, ENROLLMENT_T E, COURSE_T C
WHERE S.Student_ID = E.Student_ID AND E.Course_ID =
C.Course_ID
;GROUP BY S.Student_ID
```

النتيجة:

```
STUDENT_        GPA
-------- ----------
20001111 2.69565217
19992341      2.875
19994512      2.125
19992020 2.47826087
20001212          4
```

16- من الطلبة الذين درسوا في المادة الدراسية (MATH101) أو المادة الدراسية (MATH102) وكان تقديرهم ممتاز (4.00) أو جيد جداً (3.00)؟

الحل:

```
SELECT FName, LName
FROM STUDENT_T, ENROLLMENT_T
WHERE STUDENT_T.Student_ID = ENROLLMENT_T.Student_ID
AND
Grade = 4 OR Grade = 3) AND)
);’MATH102‘ OR Course_ID  = ’MATH101‘(Course_ID =
```

النتيجة:

```
FNAME        LNAME
------------ ------------
Saleh        Alhamad
```

17- ما أرقام وأسماء أعضاء هيئة التدريس الذين تحتوي أسماؤهم الأولى على الحرفين (SA) في أيِّ موقع بالاسم سواء كانت الحروف بالحجم الصغير أو الكبير (Capital or small letters)؟ رتب الأسماء تصاعدياً حسب الاسم الأول وتصاعدياً داخل اسم العائلة؟

الحل:

```
SELECT FaCulty_ID, FName, LName
```

```
FROM FACULTY_T
 OR FName LIKE'%Sa%'WHERE FName LIKE
 OR'%SA%'
'%sA%' OR FName LIKE '%sa%'FName LIKE
ORDER BY LName, FName;
```

أو

```
SELECT FaCulty_ID, FName, LName
FROM FACULTY_T
'%SA%'WHERE UPPER(FName) LIKE
ORDER BY LName, FName;
```

النتيجة:

```
FACULTY_ FNAME        LNAME
-------- ------------ ------------
560      Salman       Albassam
310      Saleh        Aleesa
420      Saleh        Alghamdi
540      Salem        Alhamad
660      Saud         Alkhalifa
810      Saad         Alzhrani
```

18- ما أسماء وأرقام هواتف وأسماء أقسام أعضاء هيئة التدريس الذين تبدأ أسماؤهم الأولى بالحرف (M) أو تنتهي بالحرف (D) (بغض النظر عن حجم الحرف)؟ رتب الأسماء تصاعدياً حسب الاسم الأول وتصاعدياً داخل اسم العائلة؟

الحل:

```
SELECT LName, FName, Name
FROM FACULTY_T, DEPARTMENT_T
```

```
WHERE FACULTY_T.Department_ID =
Department_T.Department_ID AND
%D‘ OR FName LIKE ’m%‘ OR FName LIKE ’M%‘(FName LIKE
 OR’%
)’%D‘ OR FName LIKE ’%d‘ OR FName LIKE ’%d %‘FName LIKE
ORDER BY LName, FName;
```

أو:

```
SELECT LName, FName, Name
FROM FACULTY_T, DEPARTMENT_T
WHERE FACULTY_T.Department_ID =
Department_T.Department_ID AND
 OR’%D %‘ OR UPPER(FName) LIKE ’M%‘(UPPER(FName) LIKE
)’%D‘UPPER(FName) LIKE
ORDER BY LName, FName;
```

النتيجة:

```
LNAME        FNAME        NAME
------------ ------------ ------------------------------
Alhamad      Mohammed     Computer Science
Alhamid      Fahad        Mathematics
Alkhalifa    Saud         Statistics
Almazid      Mishal       Physics
Alotaibi     Ahmad        Chemistry
Aloufi       Khalid       Mathematics
Alsabti      Ahmad        Electrical Engineering
Alsalem      Mahmood      Physics
Alzaid       Fahad        Statistics
Alzhrani     Saad         Electrical Engineering
```

19- ما أسماء ورواتب أعضاء هيئة التدريس التابعين لقسم الحاسب الآلي بعد زيادة مرتباتهم بمقدار (10%)؟

الحل:

```
SELECT FName, LName, (Salary * 1.1)
FROM FACULTY_T
;'CS'WHERE FACULTY_T.Department_ID =
```

النتيجة:

```
FNAME        LNAME        (SALARY*1.1)
------------ ------------ ------------
Saleh        Aleesa              33000
Mohammed     Alhamad             48400
Ghanim       Alghanim            48950
Ibraheem     Alsaleh             27500
```

20- ما أسماء وتواريخ ميلاد ومرتبات أعضاء هيئة التدريس بعد رفعها بمقدار (15%) للذين ولدوا قبل (01-07-1965) ومرتباتهم أقل من أو تساوي (40.000)؟ رتب النتيجة تصاعدياً حسب تواريخ الميلاد.

الحل:

```
SELECT FName, LName, DOB, (Salary * 1.15)
FROM FACULTY_T
 AND Salary <= 40000'WHERE FACULTY_T.DOB < '01-07-1965
ORDER BY DOB;
```

النتيجة:

FNAME	LNAME	DOB	(SALARY*1.15)
Khalid	Aloufi	22/05/63	40250
Yahya	Khorshid	12/03/65	42205

21- ما أسماء وتواريخ ميلاد ومرتبات أعضاء هيئة التدريس بعد رفعها بمقدار (15%) للذين ولدوا بعد (1970-07-01)، ومرتباتهم أكبر من (40.000)؟ رتِّب النتيجة تصاعدياً حسب المرتبات بعد الرفع.

الحل:

```
SELECT FName, LName, DOB, (Salary * 1.15)
FROM FACULTY_T
 AND Salary > 40000'WHERE FACULTY_T.DOB > '01-07-1970
ORDER BY 4;
```

النتيجة:

FNAME	LNAME	DOB	(SALARY*1.15)
Fahad	Alzaid	12/05/71	50945
Saud	Alkhalifa	13/08/72	51635

22- ما أسماء أعضاء هيئة التدريس الذين تمَّ تأهيلهم لتدريس إحدى المواد الدراسية بعد تاريخ (1996-03-01)؟

الحل:

```
SELECT FName, LName
FROM FACULTY_T, QUALIFICATION_T
WHERE FACULTY_T.FaCulty_ID =
QUALIFICATION_T.FaCulty_ID AND Date_Qualified >= '01-03-
1996';
```

النتيجة:

```
FNAME        LNAME
------------ ------------
Mohammed     Alhamad
Ghanim       Alghanim
Ibraheem     Alsaleh
Mahmood      Alsalem
Mishal       Almazid
```

23- ما أكبر راتب يتقاضاه أعضاء هيئة التدريس في الجامعة؟ أظهر النتيجة تحت مُسمَّى (MAXIMUM_SALARY)؟

الحل:

```
SELECT MAX(Salary) MAXIMUM_SALARY
FROM FACULTY_T;
```

النتيجة:

```
MAXIMUM_SALARY
--------------
         45300
```

24- ما أكبر راتب يتقاضاه أعضاء هيئة التدريس في قسم الحاسب الآلي؟ أظهر النتيجة تحت مُسمَّى (MAXIMUM_SALARY).

الحل:

```
SELECT MAX(Salary)
MAXIMUM_SALARY
FROM FACULTY_T
```

```
WHERE Department_ID = 'CS';
```

النتيجة:

```
MAXIMUM_SALARY
--------------
         44500
```

25- ما عدد أعضاء هيئة التدريس العاملين في قسم الحاسب الآلي (CS)؟ وما مجموع ومتوسط مرتباتهم وأكبر وأصغر مرتب؟

الحل:

```
SELECT COUNT(*), SUM(Salary), AVG(Salary), MAX(Salary),
MIN(Salary)
FROM FACULTY_T
;'CS'WHERE Department_ID =
```

النتيجة:

```
  COUNT(*) SUM(SALARY) AVG(SALARY) MAX(SALARY) MIN(SALARY)
---------- ----------- ----------- ----------- -----------
         4      143500       35875       44500       25000
```

26- ما مجموع ومتوسط رواتب أعضاء هيئة التدريس الذين يعملون في قسم الرياضيات (MATH)؟ أظهر النتيجة تحت مُسمَّى (SUM_SALARY) ومُسمَّى (AVG_SALARY).

الحل:

```
SELECT SUM(Salary) SUM_SALARY, AVG(Salary)
AVG_SALARY
```

```
FROM FACULTY_T

WHERE Department_ID = ‘MATH’;
```

النتيجة:

```
SUM_SALARY AVG_SALARY
---------- ----------
     60900      30450
```

-27

ما عدد أعضاء هيئة التدريس الذين لا يعملون في قسم الحاسب الآلي وتتراوح مرتباتهم ما بين (30.000) و (40.000)، وما مجموع مرتباتهم؟ أظهر عدد أعضاء هيئة التدريس تحت مُسمَّى (No_of_FaCulty).

الحل:

```
SELECT COUNT(*) No_of_FaCulty, SUM(Salary)

FROM FACULTY_T

 AND Salary >= 30000 AND‘CS’WHERE Department_ID <>
Salary <= 40000;
```

النتيجة:

```
NO_OF_FACULTY SUM(SALARY)
------------- -----------
            7      245200
```

-28 ما مجموع ومتوسط مرتبات أعضاء هيئة التدريس الذين يعملون في قسم الحاسب الآلي قبل وبعد رفعها بمقدار (10%)؟ أظهر المجموع قبل الزيادة تحت مُسمَّى

(Sum_Before_InCrease) والمجموع بعد الزيادة تحت مُسمًّى (Sum_After_InCrease)، والمتوسط قبل الزيادة تحت مُسمًّى (Avg_Before_InCrease)، والمتوسط بعد الزيادة تحت مسمى (Avg_After_InCrease).

الحل:

```
SELECT SUM(Salary) Sum_Before_InCrease, SUM(Salary * 1.1)
Sum_After_InCrease, AVG(Salary) Avg_Before_InCrease, AVG(Salary
* 1.1) Avg_After_InCrease
FROM FACULTY_T
;'CS'WHERE Department_ID =
```

النتيجة:

SUM_BEFORE_INCREASE	SUM_AFTER_INCREASE	AVG_BEFORE_INCREASE	AVG_AFTER_INCREASE
143500	157850	35875	39462.5

29- لكلِّ قسم من أقسام الجامعة، ما أكبر وأصغر ومتوسط مرتبات أعضاء هيئة التدريس في القسم؟

الحل:

```
SELECT Department_ID, MAX(Salary), MIN(Salary), AVG(Salary)
FROM FACULTY_T
GROUP BY Department_ID;
```

النتيجة:

DEPART	MAX(SALARY)	MIN(SALARY)	AVG(SALARY)
CHEM	44600	33900	39250
ENGL	40000	33800	36833.3333
PHYS	43300	29800	35000
EE	45300	33900	41133.3333
MATH	35000	25900	30450
CS	44500	25000	35875
STAT	44900	27800	39000

30- ما عدد الطلبة المُسجَّلين في كلِّ مادة دراسية على حِدَة؟

الحل:

```
SELECT Course_ID, COUNT(*) NO_OF_STUDENTS
FROM ENROLLMENT_T
GROUP BY Course_ID;
```

النتيجة:

COURSE_	NO_OF_STUDENTS
CHEM101	5
CS101	2
CS102	2
ENGL101	4
ENGL102	2
MATH101	4
MATH102	2
STAT101	2

31- ما عدد الطلبة المُسجَّلين في كلِّ مادة دراسية من المواد التي ينفذها قسم الحاسب الآلي (CS)؟

الحل:

```
SELECT Course_ID, COUNT(*)
NO_OF_STUDENTS
FROM ENROLLMENT_T
WHERE Course_ID LIKE ‘CS%’
```

```
GROUP BY Course_ID;
```

أو

```
SELECT Course_ID, COUNT(*)
NO_OF_STUDENTS
FROM ENROLLMENT_T
WHERE Course_ID IN
(SELECT Course_ID
FROM COURSE_T
CS')WHERE Department_ID  = '
GROUP BY Course_ID;
```

النتيجة:

```
COURSE_ NO_OF_STUDENTS
------- --------------
CS101                2
CS102                2
```

32- ما رقم كلِّ مادة دراسية سجل فيها أربعة طلبة؟ وما متوسط التقديرات فيها؟

الحل:

```
SELECT Course_ID, AVG(Grade)
AVERAGE_GRADE
FROM ENROLLMENT_T EN1
WHERE (SELECT COUNT(Student_ID)
FROM ENROLLMENT_T
WHERE Course_ID = EN1.Course_ID) = 4
```

```
GROUP BY Course_ID;
```

النتيجة:

```
COURSE_ AVERAGE_GRADE
------- -------------
MATH101          1.75
ENGL101          3.75
```

33- ما مجموع مرتبات أعضاء هيئة التدريس بعد رفعها بنسبة (15%) في كلِّ قسم من أقسام الجامعة عدا قسم الرياضيات (MATH)؟ رتب النتيجة تصاعدياً حسب المجموع.

الحل:

```
SELECT Department_ID, SUM(Salary *
1.15)
FROM FACULTY_T
'MATH'WHERE Department_ID <>
GROUP BY Department_ID
ORDER BY 2;
```

النتيجة:

```
DEPART SUM(SALARY*1.15)
------ ----------------
CHEM              90275
PHYS             120750
ENGL             127075
STAT             134550
EE               141910
CS               165025
```

34- ما أرقام المواد الدراسية التي نفذت من خلال أكثر من مجموعة (أو شعبة)؟ رتب النتيجة تصاعدياً حسب رمز المادة الدراسية.

الحل:

```
SELECT Course_ID
FROM SECTION_T
GROUP BY Course_ID
HAVING COUNT(Course_ID) > 1;
```

النتيجة:

```
COURSE_
-------
CHEM101
CS101
```

35- ما أسماء أعضاء هيئة التدريس وأرقام المواد الدراسية التي درسوا المجموعة الأولى منها (في أيِّ برنامج كان) خلال الفصل الدراسي (SPRING) من عام (2000)؟

الحل:

```
SELECT FName, LName, Course_ID
FROM FACULTY_T, SECTION_T
WHERE FACULTY_T.FaCulty_ID =
SECTION_T.FaCulty_ID AND SeCtion_NO
= 1
AND Semester = ‘SPRING’ AND Year =
2000;
```

النتيجة:

FNAME	LNAME	COURSE_
Khalid	Aloufi	MATH102
Mohammed	Alhamad	CS103
Mohammed	Alhamad	CS102
Salem	Alhamad	ENGL102
Turki	Alturki	STAT101

36- ما أرقام الطلبة وتقديراتهم وأرقام المواد الدراسية التي درسوها وحصلوا على تقديرات 3 أو 4؟ رتب النتيجة تنازلياً حسب التقديرات مرتبة بشكلٍ تنازلي داخل أرقام المواد الدراسية التي درسوها.

الحل:

```
SELECT Student_ID, Course_ID, Grade
FROM ENROLLMENT_T
WHERE Grade = 3 OR Grade = 4
ORDER BY Course_ID DESC, Grade DESC;
```

النتيجة:

STUDENT_	COURSE_	GRADE
20001111	STAT101	3
19992020	MATH101	3
20001111	ENGL102	4
20001111	ENGL101	4
19994512	ENGL101	4
19992341	ENGL101	4
19992020	ENGL101	3
20001111	CS102	4
19992020	CS102	3
20001111	CS101	4
20001212	CHEM101	4
19992020	CHEM101	4
19994512	CHEM101	3
19992341	CHEM101	3

37- ما رقم المادة الدراسية ومتوسط التقديرات وعدد الطلاب لكلِّ مادة سجل فيها أكثر من طالبين؟

الحل:

```
SELECT Course_ID, AVG(Grade)
AVERAGE_GRADE, COUNT(Student_ID)
NO_OF_STUDENTS
FROM ENROLLMENT_T
GROUP BY Course_ID
HAVING COUNT(Course_ID) > 2;
```

أو

```
SELECT Course_ID, AVG(Grade)
AVERAGE_GRADE, COUNT(Student_ID)
NO_OF_STUDENTS
FROM ENROLLMENT_T EN1
WHERE (SELECT COUNT(Student_ID)
FROM ENROLLMENT_T
WHERE Course_ID = EN1.Course_ID) > 2
GROUP BY Course_ID;
```

النتيجة:

```
COURSE_ AVERAGE_GRADE NO_OF_STUDENTS
------- ------------- --------------
CHEM101             3              5
MATH101          1.75              4
ENGL101          3.75              4
```

38- استخدم تعليمة إنشاء جدول (CREATE TABLE) لنسخ جدول الأقسام الدراسية (DEPARTMENT_T) بمُسمَّى (MY_DEPARTMENT_T).

الحل:

```
CREATE TABLE MY_DEPARTMENT_T
AS
SELECT *
FROM DEPARTMENT_T;
```

39- استخدم تعليمة (ALTER) لإضافة الحقل (أو العمود) (LoCation) للجدول الجديد الذي قمت بإنشائه بمُسمَّى (MY_DEPARTMENT_T)؛ بحيث تكون بياناته نصية (Text) وطولها عشر خانات.

الحل:

```
ALTER TABLE MY_DEPARTMENT_T
ADD
LoCation CHAR(10);
```

40- استخدم تعليمة (UPDATE) لإضافة بيانات العمود (LOCATION) حسب التالي:

اسم القسم (Department Name)	الموقع (LoCation)
قسم الحاسب الآلي (CS)	الرياض (RIYADH)
قسم الرياضيات (MATH)	الرياض (RIYADH)
قسم الإحصاء (STAT)	جدة (JEDDAH)
قسم الهندسة الكهربائية (EE)	الدمام (DAMMAM)
قسم الكيمياء (CHEM)	الجوف (JOUF)

جازان (JAZAN)	قسم اللغة الإنجليزية (ENGL)
جدة (JEDDAH)	قسم الفيزياء (PHYS)

الحل:

تحديث بيانات الأقسام الدراسية في الجدول الجديد

```
UPDATE MY_DEPARTMENT_T SET Location = 'RIYADH' WHERE Department_ID = 'CS';
UPDATE MY_DEPARTMENT_T SET Location = 'RIYADH' WHERE Department_ID = 'MATH';
UPDATE MY_DEPARTMENT_T SET Location = 'JEDDAH' WHERE Department_ID = 'STAT';
UPDATE MY_DEPARTMENT_T SET Location = 'DAMMAM' WHERE Department_ID = 'EE';
UPDATE MY_DEPARTMENT_T SET Location = 'JOUF' WHERE Department_ID = 'CHEM';
UPDATE MY_DEPARTMENT_T SET Location = 'JAZAN' WHERE Department_ID = 'ENGL';
UPDATE MY_DEPARTMENT_T SET Location = 'JEDDAH' WHERE Department_ID = 'PHYS';
```

41- أنشئ منظوراً (View) بمُسمَّى (COURSE_LIST_V) يدمج الحقول (Course_ID) و(Title) من جدول المواد الدراسية (COURSE_T) مع الحقل (Name) والحقل (LoCation) من الجدول الجديد الذي قمت بإنشائه (MY_DEPARTMENT_T).

الحل:

```
CREATE VIEW COURSE_LIST_V AS
SELECT Course_ID, Title, Name, LoCation
FROM COURSE_T, MY_DEPARTMENT_T
WHERE COURSE_T.Department_ID =
MY_DEPARTMENT_T.Department_ID;
```

النتيجة:

COURSE_	TITLE	NAME	LOCATION
CHEM101	CHEMISTRY (I)	Chemistry	JOUF
CHEM102	CHEMISTRY (II)	Chemistry	JOUF
CS101	JAVA PROGRAMMING	Computer Science	RIYADH
CS102	SOFTWARE ENGINEERING	Computer Science	RIYADH
CS103	C/C++ PROGRAMMING	Computer Science	RIYADH
CS104	COMPUTER ARCHITECTURE	Computer Science	RIYADH
CS105	INTRODUCTION TO DATABASE SYSTEMS	Computer Science	RIYADH
EE101	ELECTRIC CIRCUITS	Electrical Engineering	DAMMAM
EE102	ELECTRONICS (I)	Electrical Engineering	DAMMAM
EE103	ELECTRONICS (II)	Electrical Engineering	DAMMAM
EE104	COMMUNICATION NETWORKS	Electrical Engineering	DAMMAM
ENGL101	ENGLISH GRAMMAR	English Language	JAZAN
ENGL102	ENGLISH WRITING	English Language	JAZAN
ENGL103	TECHNICAL WRITING	English Language	JAZAN
MATH101	INTRODUCTION To MATHEMATICS	Mathematics	RIYADH
MATH102	DIFFERENTIAL EQUATIONS	Mathematics	RIYADH
MATH103	CALCULUS (I)	Mathematics	RIYADH
MATH104	CALCULUS (II)	Mathematics	RIYADH
MATH106	ALGEBRA	Mathematics	RIYADH
MATH107	COMPUTER MATHEMATICS	Mathematics	RIYADH
PHYS101	PHYSICS (I)	Physics	JEDDAH
PHYS102	PHYSICS (II)	Physics	JEDDAH
STAT101	INTRODUCTION TO STATISTICS	Statistics	JEDDAH
STAT102	ADVANCED STATISTICS	Statistics	JEDDAH

42- ما أرقام المواد الدراسية (Course_ID) وأسماؤها (Title) التي تُنفَّذ من خلال أقسام علمية مواقعها (LoCation) في مدينة الرياض (RIYADH) أو الجوف (JOUF)؛ وذلك حسب ورودها في المنظور (COURSE_LIST_V)؟

الحل:

```
SELECT Course_ID, Title
FROM COURSE_LIST_V
WHERE LoCation = ‘RIYADH’ OR LoCation =
‘JOUF’;
```

النتيجة:

```
COURSE_ TITLE
------- -----------------------------------
CHEM101 CHEMISTRY (I)
CHEM102 CHEMISTRY (II)
CS101   JAVA PROGRAMMING
CS102   SOFTWARE ENGINEERING
CS103   C/C++ PROGRAMMING
CS104   COMPUTER ARCHITECTURE
CS105   INTRODUCTION TO DATABASE SYSTEMS
MATH101 INTRODUCTION To MATHEMATICS
MATH102 DIFFERENTIAL EQUATIONS
MATH103 CALCULUS (I)
MATH104 CALCULUS (II)
MATH106 ALGEBRA
MATH107 COMPUTER MATHEMATICS
```

43- أنشئ فهرساً للجدول الجديد (MY_DEPARTMENT_T) على مواقع الأقسام الدراسية (LOCATION) باسم (DEPARTMENT_LOCATION_IDX).

الحل:

```
CREATE INDEX
DEPARTMENT_LOCATION_IDX ON
MY_DEPARTMENT_T(LoCation);
```

ملحق رقم (2)

حالة دراسية – قاعدة بيانات شركة عقارية افتراضية

ملحق رقم (2) – 1: قواعد العمل المعمول بها في الشركة العقارية:

تعتزمُ إحدى الشركات العقارية الكبرى مكننة عملها الإداري؛ من خلال تطوير نظم تطبيقية تعتمد في بنائها على نظم قواعد البيانات. وكخطوة أولى تعتزم الشركة نمذجة بعض قواعد العمل المعمول بها في الشركة والتي تحكم طبيعة عملها باستخدام النموذج المفاهيمي «كينونة - علاقة». باستخدام قواعد العمل التالية، المطلوب هو تصميم قاعدة البيانات باستخدام نموذج «كينونة - علاقة».

1- يُوجَد للشركة مجموعةٌ من المكاتب (OffiCes) في مناطق ومدن مختلفة، ولكل مكتب رمز (OffiCe_ID) يميّزه بشكلٍ منفرد عن بقية المكاتب التابعة للشركة وعنوان (Address) ورقم هاتف (Telephone_No).

2- يعمل في الشركة مجموعةٌ من الموظفين، ولكلِّ موظف رمز وظيفي (Employee_ID) يميّزه عن بقية الموظفين العاملين في الشركة، واسم يتكون من (الاسم الأول، واسم الأب، واسم العائلة)، وتاريخ ميلاد (DOB)، وعنوان سكني، ورقم هاتف، وشهادة علمية أو أكثر (ACademiC_Degree) تتكون من الدرجة العلمية (Degree)، وتاريخ الحصول عليها (Date)، ومكان الحصول عليها (Issuing_Institution).

3- يعمل كلُّ موظف في مكتب واحد فقط في أية فترة زمنية. وعندما يبدأ الموظف في العمل في مكتب ما؛ فإن هنالك تاريخاً لبدئه للعمل في المكتب (Starting_Date). كما أنه قد

يعمل الموظف في أكثر من مكتب؛ ولكن في فترات زمنية مختلفة وغير متداخلة مع بعضها. ويعمل في المكتب الواحد موظفٌ واحدٌ أو أكثر.

4- تشرف الشركة على مجموعة من العقارات ولكلِّ عقار رمز (Estate_ID) يميِّزه بشكلٍ منفرد عن بقية العقارات التي تشرف عليها الشركة، وموقع (Address). والعقار الواحد الذي تشرف عليه الشركة؛ يجب أن يكون إمَّا للإيجار (Rental_Estate)، أو للبيع (Sale_Estate). وعندما يكون العقار معروضاً للبيع؛ فإنه ليس من الممكن أن يكون للإيجار، كما أن العقار المعروض للإيجار لا يمكن أن يكون قابلاً للبيع. ولكلِّ عقار معروض للإيجار تاريخ (Date) يوضِّح اليوم الذي من الممكن أن يبدأ به إيجار العقار؛ بحيث إنه لا يمكِن تأجير العقار قبله، وسعرٌ لإيجاره السنوي (Yearly_Rent)، ومبلغٌ للتأمين عليه (InsuranCe_Amount). أمَّا العقار المعروض للبيع؛ فله سعر (PriCe) ونسبة لعمولة الشركة من مبلغ البيع، وهي 5.2% من سعر بيع العقار.

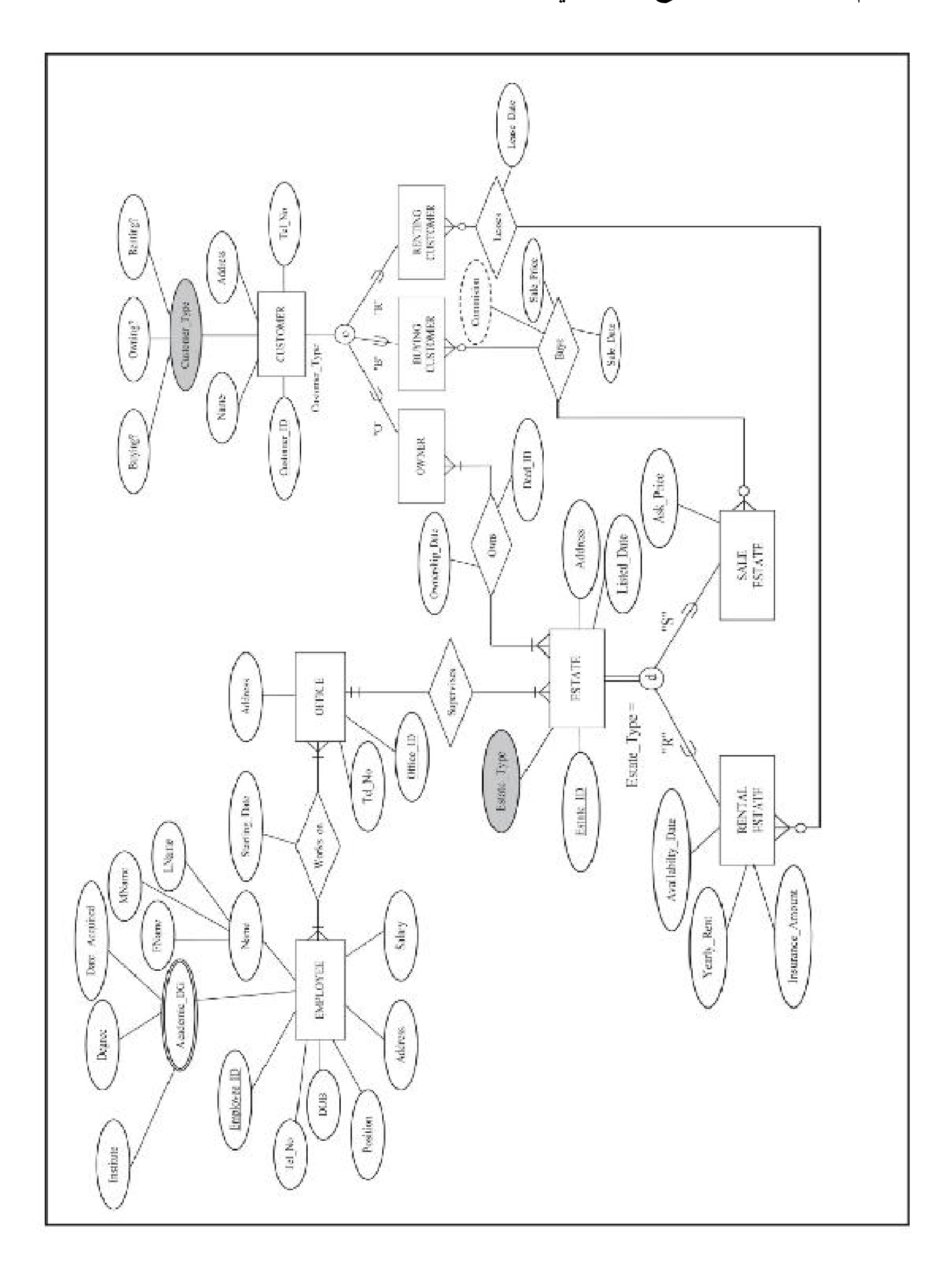
5- يُشرِف كلُّ مكتب من مكاتب الشركة على واحدٍ أو أكثر من العقارات (سواءً كانت معروضة للبيع أو للإيجار)، ويجب أن يتمَّ الإشراف على العقار من قِبل مكتب واحد فقط من مكاتب الشركة.

6- يتعامل مع الشركة مجموعةٌ من العملاء (Customers). ولكلِّ عميلٍ رمزٌ يميِّزه عن بقية العملاء (Customer_ID)، واسم، ورقم تليفون، وعنوان بريدي. والعميل الواحد إمَّا أن يكون مالكاً لعقار (يرغب في بيعه أو في إيجاره) (Owner) أو مشترياً لعقار (Buying_Customer) أو مستأجراً لعقار (Renting_Customer).

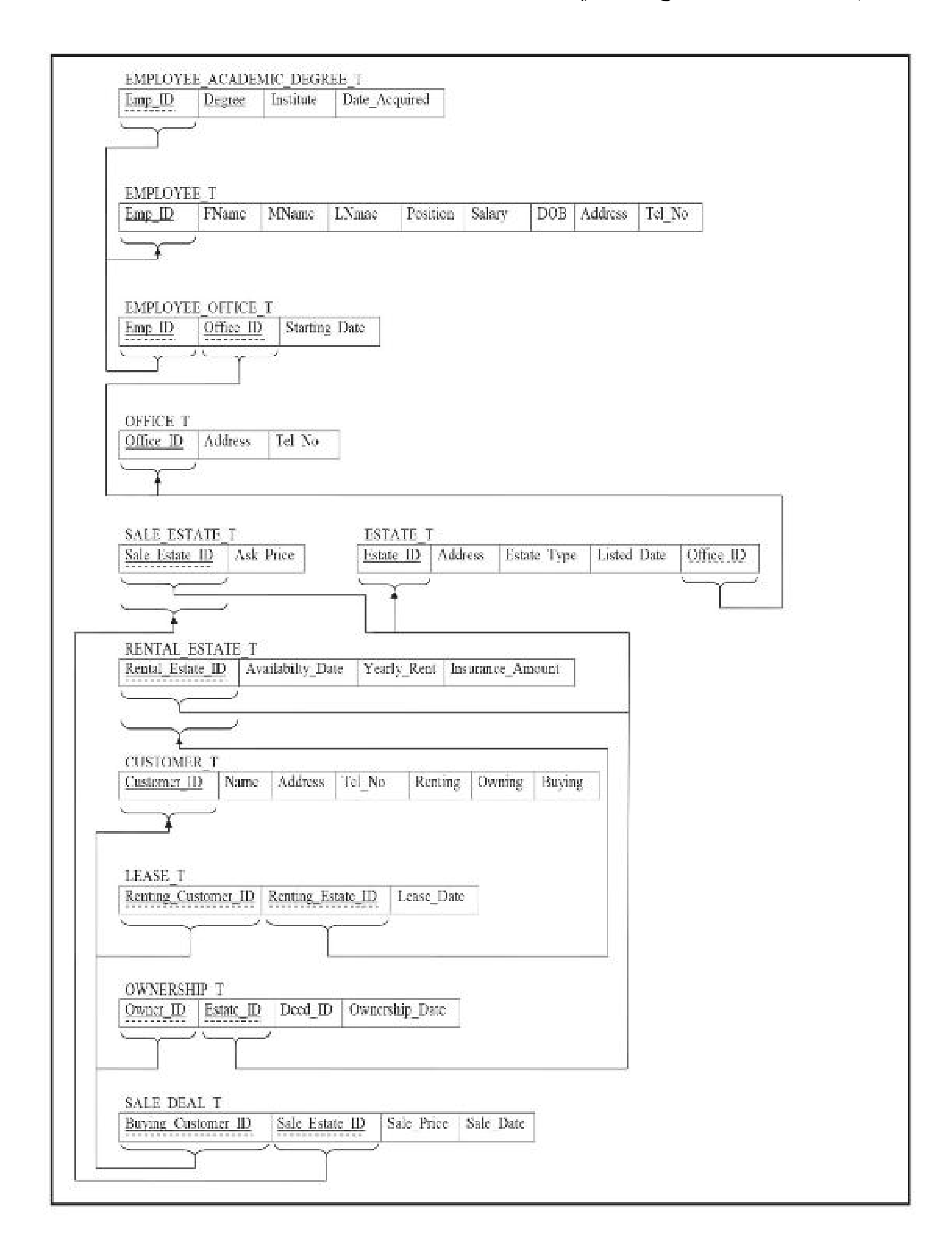
7- يجب أن يملك (Owns) كلَّ عقار واحدٌ أو أكثر من المالكين، والمالك الواحد قد يكون له واحدٌ أو أكثر من العقارات (المعروضة للبيع أو للإيجار). ولكلِّ مالك لعقار تاريخ (Date) يبيِّن ملكيته له ورقم الصك (Estate_Ownership_ID) الذي تملك من خلاله العقار. أمَّا العقار المعروض للبيع فله صفر أو أكثر من المشترين، ولكلِّ مشترٍ صفر أو أكثر من العقارات المشتراة من الشركة.

8- عندما يتم بيع عقار لأحد المشترين، فإن لعملية البيع تاريخ (Date) وسعر للبيع(Sale_PriCe) قد يكون مختلفاً عن السعر الذي تم تحديده بشكل مبدئي كقيمة للعقار. أما العقار المعروض للإيجار فيتم إيجاره وفق عقد (Lease_Agreement) يتحدد فيه المستأجر والعقار، كما يتضمن العقد تاريخ بدء سريان عقد الإيجار (Lease_Date) الذي قد يكون مختلفاً عن التاريخ الذي عرض فيه العقار للإيجار.

ملحق رقم (2) – 2: النموذج المفاهيمي لقاعدة بيانات الشركة العقارية:



ملحق رقم (2) – 3: النموذج المنطقي لقاعدة بيانات الشركة العقارية:



ملحق رقم(2) – 4: إنشاء قاعدة البيانات باستخدام تعليمات (SQL) في بيئة أوراكل (SQL*Plus):

– جدول الموظفين (EMPLOYEE_T):

إنشاء جدول الموظفين

```
CREATE TABLE EMPLOYEE_T
        (EMP_ID         NUMBER          NOT NULL,
        FNAME           CHAR(12)        NOT NULL,
        MNAME           CHAR(12)        NOT NULL,
        LNAME           CHAR(12)        NOT NULL,
        Position        CHAR(17)        NOT NULL,
        Salary          Number          NOT NULL,
        DOB             DATE,
        ADDRESS         CHAR(35),
        TEL_NO          CHAR(15),
CONSTRAINT EMPLOYEE_T_PK PRIMARY KEY (EMP_ID));
```

يتمُّ إدخال البيانات للجدول باستخدام (SQL) حسب التعليمات التالية:

إدخال بيانات الموظفين

```
INSERT INTO EMPLOYEE_T VALUES (1010, 'Saleh', 'Ahmad', 'Alhamad', 'Director General', 35000,
'22/05/1963', '13 Almutanabi Street, Riyadh', '009661-454-2341');
INSERT INTO EMPLOYEE_T VALUES (1020, 'Abdullah', 'Abdulrahman', 'Aloufi', 'Sale Person', 22000,
'07/10/1970', '25 Jareer Street, Riyadh', '009661-456-7733');
INSERT INTO EMPLOYEE_T VALUES (1030, 'Salem', 'Mohsen', 'Algamdi', 'Sale Person', 20000,
'13/09/1966', '98 Bin Taimiah Street, Jeddah', '009662-454-5412');
INSERT INTO EMPLOYEE_T VALUES (1040, 'Mishal', 'Hamad', 'Alyousef', 'Sale Person', 21500,
'12/08/1969', '13 Alsouk Street, Taif', '009665-050-3131');
INSERT INTO EMPLOYEE_T VALUES (1050, 'Khalid', 'Saud', 'Alsultan', 'Office Manager', 30000,
'22/06/1964', '22 Bin Hamdan Street, Jeddah', '009665-550-2015');
INSERT INTO EMPLOYEE_T VALUES (1060, 'Mohammed', 'Salman', 'Abdelaleem',
'Office Manager', 31500, '27/08/1972', '10 Bin Hamdan Srreer, Riyadh', '009665-555-2345');
INSERT INTO EMPLOYEE_T VALUES (1070, 'Sami', 'Suliman', 'Aloutaibi', 'Office Manager', 30000,
'01/11/1967', '67 Alfadel Street, Dammam', '009665-050-6178');
INSERT INTO EMPLOYEE_T VALUES (1080, 'Saud', 'Ahmad', 'Alganim', 'Office Manager', 20000,
'13/12/1972', '24 Alfadel Street, Aljouf', '009665-550-2134');
INSERT INTO EMPLOYEE_T VALUES (1090, 'Abdulrahman', 'Mansour', 'Abdulsalam',
'Sale Person', 22000, '15/09/1975', '10 Almadinah Street, Skaka', '009665-550-7891');
INSERT INTO EMPLOYEE_T VALUES (2010, 'Salman', 'Mohammed', 'Alsaleh',
'Sale Person', 22000, '26/3/1976', '15 King Fahad Road, Dammam', '009665-050-3231');
INSERT INTO EMPLOYEE_T VALUES (2020, 'Khalid', 'Mohammed', 'Alomar',
'Sale Person', 22000, '18/05/1961', '91 Alwadi Street, Najran', '009665-050-1516');
INSERT INTO EMPLOYEE_T VALUES (2030, 'Minwer', 'Hamad', 'Almutairi',
'Sale Person', 22000, '08/11/1965', '87 Alhamra Road, Jizan', '009665-056-1231');
INSERT INTO EMPLOYEE_T VALUES (2040, 'Turki', 'Khalid', 'Alassaf',
'Sale Person', 22000, '15/10/1969', '25 Prince Abdullah Street, Riyadh', '009665-057-8796');
INSERT INTO EMPLOYEE_T VALUES (2050, 'Saleh', 'Hamad', 'Alzaid', 'Sale Person', 22000, '13/07/1971',
'25 King Faisal Srreer, Riyadh', '009665-057-7788');
INSERT INTO EMPLOYEE_T VALUES (2060, 'Ghanim', 'Abdullah', 'Alhmoud', 'Sale Person', 22000,
'21/02/1969', '56 Altahliah Street, Jeddah', '009665-055-9781');
INSERT INTO EMPLOYEE_T VALUES (2070, 'Sultan', 'Saleh', 'Abdulgader', 'Sale Person', 22000,
'29/04/1975', '123 Salman Alfarsi Street, Riyadh', '009665-059-3831');
INSERT INTO EMPLOYEE_T VALUES (2080, 'Suliman', 'Abdullah', 'Almushari', 'Sale Person', 22000,
'05/03/1973', '45 Prince Sultan Streer, Najran', '009665-056-3111');
INSERT INTO EMPLOYEE_T VALUES (2090, 'Ahmad', 'Abdullah', 'Alsaif', 'Sale Person', 22000,
'18/04/1968', '13 Khalifa Street, Taif', '009665-556-1231');
INSERT INTO EMPLOYEE_T VALUES (3010, 'Ahmad', 'Mansour', 'Alshemamri', 'Sale Person', 22000,
'19/09/1975', '15 Othman street, Jizan', '009665-055-3486');
INSERT INTO EMPLOYEE_T VALUES (3020, 'Mohammed', 'Khalid', 'Alzamil', 'Sale Person', 22000,
'17/10/1961', '67 Abubaker Road, Abha', '009665-057-6578');
INSERT INTO EMPLOYEE_T VALUES (3030, 'Mansour', 'Abdullah', 'Alzamil',
'Office Manager', 22000, '03/12/1960', '13 King Abdulaziz Road, Tabouk', '009665-054-3331');
```

– جدول الشهادات العلمية للموظفين (**EMPLOYEE_ACADEMIC_DEGREE_T**):

إنشاء جدول الشهادات العلمية للموظفين

```
CREATE TABLE EMPLOYEE_ACADEMIC_DEGREE_T
        (EMP_ID              NUMBER          NOT NULL,
        DEGREE               CHAR(10)        NOT NULL,
        INSTITUTE            CHAR(30)        NOT NULL,
        DATE_ACQUIRED        DATE            NOT NULL,
        CONSTRAINT EMPLOYEE_ACADEMIC_DEGREE_T_PK
                PRIMARY KEY (EMP_ID, DEGREE),
        CONSTRAINT EMPLOYEE_ACADEMIC_DEGREE_T_FK
        FOREIGN KEY (EMP_ID) REFERENCES EMPLOYEE_T(EMP_ID));
```

يتمُّ إدخال البيانات للجدول باستخدام (SQL) حسب التعليمات التالية:

إدخال بيانات الشهادات العلمية للموظفين

```
INSERT INTO EMPLOYEE_ACADEMIC_DEGREE_T VALUES (1010, 'BA', 'Louisiana State University',
'01/05/1998');
INSERT INTO EMPLOYEE_ACADEMIC_DEGREE_T VALUES (1010, 'MBA', 'Louisiana State University',
'12/01/2000');
INSERT INTO EMPLOYEE_ACADEMIC_DEGREE_T VALUES (1020, 'BS', 'University of Pittsburgh',
'11/03/1999');
INSERT INTO EMPLOYEE_ACADEMIC_DEGREE_T VALUES (1020, 'MBA', 'University of Pittsburgh',
'17/04/2001');
INSERT INTO EMPLOYEE_ACADEMIC_DEGREE_T VALUES (1030, 'BA', 'King Saud University',
'13/11/1997');
INSERT INTO EMPLOYEE_ACADEMIC_DEGREE_T VALUES (1040, 'BA', 'King Abdulaziz University',
'27/02/2000');
INSERT INTO EMPLOYEE_ACADEMIC_DEGREE_T VALUES (1050, 'BA', 'Arkansas State University',
'12/03/2000');
INSERT INTO EMPLOYEE_ACADEMIC_DEGREE_T VALUES (1060, 'BA', 'University of Oklahoma',
'10/11/1997');
INSERT INTO EMPLOYEE_ACADEMIC_DEGREE_T VALUES (1070, 'BA', 'King Saud University',
'27/10/1998');
INSERT INTO EMPLOYEE_ACADEMIC_DEGREE_T VALUES (1080, 'BA', 'King Saud University',
'27/10/1998');
INSERT INTO EMPLOYEE_ACADEMIC_DEGREE_T VALUES (1080, 'MBA', 'King Saud University',
'16/12/2000');
INSERT INTO EMPLOYEE_ACADEMIC_DEGREE_T VALUES (1090, 'BA', 'King Abdulaziz University',
'21/11/1996');
INSERT INTO EMPLOYEE_ACADEMIC_DEGREE_T VALUES (2010, 'BA', 'Philadelphia State University',
'10/08/2000');
INSERT INTO EMPLOYEE_ACADEMIC_DEGREE_T VALUES (2020, 'BA', 'King Abdulaziz University',
'27/02/2000');
INSERT INTO EMPLOYEE_ACADEMIC_DEGREE_T VALUES (2030, 'BA', 'King Abdulaziz University',
'27/02/2000');
INSERT INTO EMPLOYEE_ACADEMIC_DEGREE_T VALUES (2040, 'BA', 'King Faisal University',
'03/02/2000');
INSERT INTO EMPLOYEE_ACADEMIC_DEGREE_T VALUES (2050, 'BA', 'King Faisal University',
'03/02/2000');
INSERT INTO EMPLOYEE_ACADEMIC_DEGREE_T VALUES (2060, 'BA', 'University of Oklahoma',
'07/06/2000');
INSERT INTO EMPLOYEE_ACADEMIC_DEGREE_T VALUES (2070, 'BA', 'Philadelphia State University',
'10/08/2000');
INSERT INTO EMPLOYEE_ACADEMIC_DEGREE_T VALUES (2080, 'BA', 'King Saud University',
'28/11/2001');
INSERT INTO EMPLOYEE_ACADEMIC_DEGREE_T VALUES (2090, 'BA', 'Philadelphia State University',
'10/08/2000');
INSERT INTO EMPLOYEE_ACADEMIC_DEGREE_T VALUES (3010, 'BA', 'King Faisal University',
'03/02/1999');
INSERT INTO EMPLOYEE_ACADEMIC_DEGREE_T VALUES (3020, 'BA', 'King Faisal University',
'03/02/1999');
INSERT INTO EMPLOYEE_ACADEMIC_DEGREE_T VALUES (3030, 'BA', 'King Saud University',
'28/11/2001');
```

- **جدول مكاتب الشركة (OFFICE_T):**

إنشاء جدول مكاتب الشركة

```
CREATE TABLE OFFICE_T
        (OFFICE_ID          NUMBER          NOT NULL,
        ADDRESS             CHAR(65)        NOT NULL,
        TEL_NO              CHAR(15)        NOT NULL,
        CONSTRAINT OFFICE_T_PK PRIMARY KEY (OFFICE_ID));
```

يتمُّ إدخال البيانات للجدول باستخدام (SQL) حسب التعليمات التالية:

إدخال بيانات مكاتب العقارات

```
INSERT INTO OFFICE_T VALUES (1, 'Head Office, Al-Olya Street, Riyadh, SA',
'009661-272-4141');
INSERT INTO OFFICE_T VALUES (2, 'Riyadh Main Office, Al-Farazdaq Street, Al-
Malaz, Riyadh, SA', '009661-474-3030');
INSERT INTO OFFICE_T VALUES (3, 'Jeddah Main Office, Prince Mohammed Bin
Saud Street, Jeddah, SA', '009662-272-4141');
INSERT INTO OFFICE_T VALUES (4, 'Dammam Main Office, King Fahad Street,
Dammam, SA', '009663-231-4141');
INSERT INTO OFFICE_T VALUES (5, 'Jouf Office, Central Street, Al-Jouf, SA', '009664-
521-4141');
```

- **جدول مكاتب عمل الموظفين (EMPLOYEE_OFFICE_T):**

إنشاء جدول مكاتب عمل الموظفين

```
CREATE TABLE EMPLOYEE_OFFICE_T
        (EMP_ID                 NUMBER          NOT NULL,
        OFFICE_ID               NUMBER          NOT NULL,
        STARTING_DATE           DATE            NOT NULL,
        CONSTRAINT EMPLOYEE_OFFICE_T_PK
                PRIMARY KEY (EMP_ID, OFFICE_ID),
        CONSTRAINT EMPLOYEE_OFFICE_T_FK1
                FOREIGN KEY (EMP_ID) REFERENCES EMPLOYEE_T(EMP_ID),
        CONSTRAINT EMPLOYEE_OFFICE_T_FK2
                FOREIGN KEY (OFFICE_ID) REFERENCES OFFICE_T(OFFICE_ID));
```

يتمُّ إدخال البيانات للجدول باستخدام (SQL) حسب التعليمات التالية:

إدخال بيانات مكاتب عمل الموظفين

```
INSERT INTO EMPLOYEE_OFFICE_T VALUES (1010, 1, '29/04/2001');
INSERT INTO EMPLOYEE_OFFICE_T VALUES (1020, 1, '14/09/2002');
INSERT INTO EMPLOYEE_OFFICE_T VALUES (1030, 3, '15/04/2002');
INSERT INTO EMPLOYEE_OFFICE_T VALUES (1040, 3, '16/02/2002');
INSERT INTO EMPLOYEE_OFFICE_T VALUES (1050, 2, '29/11/2002');
INSERT INTO EMPLOYEE_OFFICE_T VALUES (1060, 3, '02/12/2002');
INSERT INTO EMPLOYEE_OFFICE_T VALUES (1070, 4, '18/10/2002');
INSERT INTO EMPLOYEE_OFFICE_T VALUES (1080, 5, '13/09/2002');
INSERT INTO EMPLOYEE_OFFICE_T VALUES (1090, 5, '28/06/2002');
INSERT INTO EMPLOYEE_OFFICE_T VALUES (2010, 4, '21/06/2003');
INSERT INTO EMPLOYEE_OFFICE_T VALUES (2020, 3, '20/05/2002');
INSERT INTO EMPLOYEE_OFFICE_T VALUES (2030, 3, '12/02/2003');
INSERT INTO EMPLOYEE_OFFICE_T VALUES (2040, 2, '11/08/2005');
INSERT INTO EMPLOYEE_OFFICE_T VALUES (2050, 2, '17/09/2003');
INSERT INTO EMPLOYEE_OFFICE_T VALUES (2060, 3, '15/11/2004');
INSERT INTO EMPLOYEE_OFFICE_T VALUES (2070, 2, '09/12/2002');
INSERT INTO EMPLOYEE_OFFICE_T VALUES (2080, 3, '07/01/2003');
INSERT INTO EMPLOYEE_OFFICE_T VALUES (2090, 4, '20/05/2007');
INSERT INTO EMPLOYEE_OFFICE_T VALUES (1070, 1, '05/12/2001');
INSERT INTO EMPLOYEE_OFFICE_T VALUES (1080, 3, '09/11/2003');
INSERT INTO EMPLOYEE_OFFICE_T VALUES (1090, 3, '06/12/2001');
INSERT INTO EMPLOYEE_OFFICE_T VALUES (3010, 1, '08/06/2005');
INSERT INTO EMPLOYEE_OFFICE_T VALUES (3020, 3, '06/03/2002');
INSERT INTO EMPLOYEE_OFFICE_T VALUES (3030, 5, '15/02/2001');
```

– جدول العقارات (ESTATE_T):

```
إنشاء جدول العقارات

CREATE TABLE ESTATE_T
        (ESTATE_ID      NUMBER        NOT NULL,
        ADDRESS         CHAR(50)      NOT NULL,
        ESTATE_TYPE     CHAR(1)       CHECK (ESTATE_TYPE IN ('S','R')),
        LISTED_DATE     DATE          NOT NULL,
        OFFICE_ID       NUMBER                  NOT NULL,
        CONSTRAINT SALE_ESTATE_T_PK PRIMARY KEY (ESTATE_ID),
        CONSTRAINT SALE_ESTATE_T_FK
            FOREIGN KEY (OFFICE_ID) REFERENCES OFFICE_T(OFFICE_ID));
```

يتمُّ إدخال البيانات للجدول باستخدام (SQL) حسب التعليمات التالية:

```
إدخال بيانات العقارات
INSERT INTO ESTATE_T VALUES (30, '12 AlMutanbi Street, Riyadh', 'S', '10/05/2000', 2);
INSERT INTO ESTATE_T VALUES (60, '17 Alolya main Road, Riyadh', 'S', '11/08/2008', 2);
INSERT INTO ESTATE_T VALUES (90, '12 AlMutanbi Street, Jeddah', 'S', '15/07/2007', 3);
INSERT INTO ESTATE_T VALUES (120, '17 Alolya main Road, Dammam', 'S', '01/02/2002', 4);
INSERT INTO ESTATE_T VALUES (150, '12 AlMutanbi Street, Jouf', 'S', '09/07/2005', 5);
INSERT INTO ESTATE_T VALUES (180, '17 Alolya main Road, Jeddah', 'S', '21/03/2004', 3);
INSERT INTO ESTATE_T VALUES (210, '12 AlMutanbi Street, Dammam', 'S', '10/12/2002', 4);
INSERT INTO ESTATE_T VALUES (240, '17 Alolya main Road, Jouf', 'S', '05/11/1998', 5);
INSERT INTO ESTATE_T VALUES (270, '12 AlMutanbi Street, Jeddah', 'S', '04/03/1993', 3);
INSERT INTO ESTATE_T VALUES (300, '17 Alolya main Road, Riyadh', 'S', '27/06/2008', 2);
INSERT INTO ESTATE_T VALUES (330, '12 AlMutanbi Street, Jouf', 'S', '07/02/2008', 5);
INSERT INTO ESTATE_T VALUES (360, '17 Alolya main Road, Riyadh', 'S', '13/01/2004', 2);
INSERT INTO ESTATE_T VALUES (390, '12 AlMutanbi Street, Jeddah', 'S', '03/01/2003', 3);
INSERT INTO ESTATE_T VALUES (420, '17 Alolya main Road, Dammam', 'S', '26/04/2003', 4);
INSERT INTO ESTATE_T VALUES (450, '12 AlMutanbi Street, Jouf', 'S', '20/02/2009', 5);
INSERT INTO ESTATE_T VALUES (480, '17 Alolya main Road, Riyadh', 'S', '13/01/2009', 2);
INSERT INTO ESTATE_T VALUES (510, '12 AlMutanbi Street, Jeddah', 'S', '04/01/2008', 3);
INSERT INTO ESTATE_T VALUES (540, '17 Alolya main Road, Dammam', 'S', '03/11/2007', 4);
INSERT INTO ESTATE_T VALUES (570, '12 AlMutanbi Street, Jouf', 'S', '14/08/2006', 5);
INSERT INTO ESTATE_T VALUES (600, '17 Alolya main Road, Riyadh', 'S', '12/10/2008', 2);
INSERT INTO ESTATE_T VALUES (630, '12 AlMutanbi Street, Jouf', 'S', '09/11/2005', 5);
INSERT INTO ESTATE_T VALUES (660, '17 Alolya main Road, Riyadh', 'S', '12/01/2007', 2);
```

تابع - إدخال بيانات العقارات

```
INSERT INTO ESTATE_T VALUES (5, '13 Salah Eldeen AlAyoubi, Malaz, Riyadh', 'R', '12/04/2009', 2);
INSERT INTO ESTATE_T VALUES (15, '18 Ahmad bin Hanbal Street, Nassem, Riyadh', 'R', '12/01/2008', 2);
INSERT INTO ESTATE_T VALUES (25, '15 Salah Eldeen AlAyoubi, Malaz, Riyadh', 'R', '12/01/2009', 2);
INSERT INTO ESTATE_T VALUES (35, '134 Ahmad bin Hanbal Street, Nassem, Riyadh', 'R', '12/01/2010', 2);
INSERT INTO ESTATE_T VALUES (45, '113 Bin Khldoon Street, Alhamra, Jeddah', 'R', '14/01/2009', 3);
INSERT INTO ESTATE_T VALUES (55, '18 Alsafa Street, Aljazeera, Jeddah', 'R', '12/06/2007', 3);
INSERT INTO ESTATE_T VALUES (65, '123 King Abdullaziz Road, Skaka, Jouf', 'R', '22/05/2007', 5);
INSERT INTO ESTATE_T VALUES (75, '1117 Ammar Bin Yasser Road, Dammam', 'R', '27/10/2008', 4);
INSERT INTO ESTATE_T VALUES (85, '12 Almutanabi Street, Dammam', 'R', '01/01/2008', 4);
INSERT INTO ESTATE_T VALUES (95, '11 Prince Syultan Street, Jouf', 'R', '22/12/2008', 5);
INSERT INTO ESTATE_T VALUES (105, '177 Subah Aljaber, Olya, Riyadh', 'R', '25/01/2006', 2);
INSERT INTO ESTATE_T VALUES (115, '188 Eygpt Road, Albarha, Riyadh', 'R', '08/09/2007', 2);
INSERT INTO ESTATE_T VALUES (125, '138 Alyamama Road, Jeddah', 'R', '19/07/2008', 3);
INSERT INTO ESTATE_T VALUES (135, '183 Khaled Bin Alwaleed, Jeddah', 'R', '16/09/2008', 3);
INSERT INTO ESTATE_T VALUES (145, '210 Hassan Bin Thabet Street, Jeddah', 'R', '02/03/2008', 3);
INSERT INTO ESTATE_T VALUES (155, '218 Othman Bin Affan Street, Riyadh', 'R', '14/01/2006', 2);
INSERT INTO ESTATE_T VALUES (165, '213 King Fahad Road, Riyadh', 'R', '19/05/2006', 2);
INSERT INTO ESTATE_T VALUES (175, '1018 King Fahad Road, Riyadh', 'R', '14/11/2009', 2);
INSERT INTO ESTATE_T VALUES (185, '1113 Bin Khldoon Street, Alhamra, Jeddah', 'R', '13/02/2007', 3);
INSERT INTO ESTATE_T VALUES (195, '1118 Mohammed Bin Nasser Street, Dammam', 'R', '27/11/2009', 4);
INSERT INTO ESTATE_T VALUES (205, '2313 Sultan Alsaleh Street, Jouf', 'R', '11/01/2009', 5);
INSERT INTO ESTATE_T VALUES (215, '3218 King Abdullaziz Road, Riyadh', 'R', '17/10/2009', 2);
INSERT INTO ESTATE_T VALUES (225, '1371 King Fahad Road, Dammam', 'R', '26/03/2008', 4);
INSERT INTO ESTATE_T VALUES (235, '1028 King Fisal Road, Jeddah', 'R', '07/12/2008', 3);
INSERT INTO ESTATE_T VALUES (245, '1235 King Khaled Road, Riyadh', 'R', '14/12/2007', 2);
INSERT INTO ESTATE_T VALUES (255, '128 King Faisl Road, Jouf', 'R', '25/12/2006', 5);
INSERT INTO ESTATE_T VALUES (265, '143 Salman Alfaresi Street, Riyadh', 'R', '17/09/2005', 2);
INSERT INTO ESTATE_T VALUES (275, '189 King Abdulaziz Road, Dammam', 'R', '19/10/2009', 4);
INSERT INTO ESTATE_T VALUES (285, '175 Prince Salman Bin Abdulaziz Road, Riyadh', 'R', '27/08/2009', 2);
INSERT INTO ESTATE_T VALUES (295, '147 King Fahad Road, Jeddah', 'R', '16/02/2008', 3);
```

– **جدول أسعار العقارات المعروضة للبيع (SALE_ESTATE_T):**

```
إنشاء جدول أسعار العقارات المعروضة للبيع

CREATE TABLE SALE_ESTATE_T
      (SALE_ESTATE_ID              NUMBER              NOT NULL,
      ASK_PRICE                    NUMBER              NOT NULL,
      CONSTRAINT SALE_ESTATE_PK PRIMARY KEY (SALE_ESTATE_ID),
      CONSTRAINT SALE_ESTATE_FK FOREIGN KEY (SALE_ESTATE_ID)
                 REFERENCES ESTATE_T(ESTATE_ID));
```

يتمُّ إدخال البيانات للجدول باستخدام (SQL) حسب التعليمات التالية:

```
إدخال بيانات أسعار العقارات المعروضة للبيع
INSERT INTO SALE_ESTATE_T VALUES (30, 766000);
INSERT INTO SALE_ESTATE_T VALUES (60, 1100000);
INSERT INTO SALE_ESTATE_T VALUES (90, 510000);
INSERT INTO SALE_ESTATE_T VALUES (120, 2110000);
INSERT INTO SALE_ESTATE_T VALUES (150, 1100000);
INSERT INTO SALE_ESTATE_T VALUES (180, 1500000);
INSERT INTO SALE_ESTATE_T VALUES (210, 750000);
INSERT INTO SALE_ESTATE_T VALUES (240, 920000);
INSERT INTO SALE_ESTATE_T VALUES (270, 450000);
INSERT INTO SALE_ESTATE_T VALUES (300, 930000);
INSERT INTO SALE_ESTATE_T VALUES (330, 970000);
INSERT INTO SALE_ESTATE_T VALUES (360, 630000);
INSERT INTO SALE_ESTATE_T VALUES (390, 750000);
INSERT INTO SALE_ESTATE_T VALUES (420, 955000);
INSERT INTO SALE_ESTATE_T VALUES (450, 1400000);
INSERT INTO SALE_ESTATE_T VALUES (480, 2900000);
INSERT INTO SALE_ESTATE_T VALUES (510, 3100000);
INSERT INTO SALE_ESTATE_T VALUES (540, 2750000);
INSERT INTO SALE_ESTATE_T VALUES (570, 610000);
INSERT INTO SALE_ESTATE_T VALUES (600, 4300000);
INSERT INTO SALE_ESTATE_T VALUES (630, 830000);
INSERT INTO SALE_ESTATE_T VALUES (660, 980000);
```

– جدول أسعار العقارات المعروضة للإيجار (**RENTAL_ESTATE_T**):

إنشاء جدول أسعار العقارات المعروضة للإيجار

```
CREATE TABLE RENTAL_ESTATE_T
        (RENTAL_ESTATE_ID          NUMBER              NOT NULL,
        AVAILABILITY_DATE          DATE                NOT NULL,
        YEARLY_RENT                NUMBER              NOT NULL,
        INSURANCE_AMOUNT           NUMBER              NOT NULL,
        CONSTRAINT RENTAL_ESTATE_PK PRIMARY KEY (RENTAL_ESTATE_ID),
        CONSTRAINT RENTAL_ESTATE_FK FOREIGN KEY (RENTAL_ESTATE_ID)
                REFERENCES ESTATE_T(ESTATE_ID));
```

يتمُّ إدخال البيانات للجدول باستخدام (SQL) حسب التعليمات التالية:

إدخال بيانات أسعار العقارات المعروضة للإيجار

```
INSERT INTO RENTAL_ESTATE_T VALUES (5, '01/05/2010', 21000, 3000);
INSERT INTO RENTAL_ESTATE_T VALUES (15, '01/12/2010', 25000, 4000);
INSERT INTO RENTAL_ESTATE_T VALUES (25, '01/05/2011', 25000, 3000);
INSERT INTO RENTAL_ESTATE_T VALUES (35, '01/04/2011', 35000, 5000);
INSERT INTO RENTAL_ESTATE_T VALUES (45, '01/07/2010', 31000, 5000);
INSERT INTO RENTAL_ESTATE_T VALUES (55, '01/12/2009', 35000, 5000);
INSERT INTO RENTAL_ESTATE_T VALUES (65, '01/05/2010', 41000, 10000);
INSERT INTO RENTAL_ESTATE_T VALUES (75, '01/08/2009', 35000, 5000);
INSERT INTO RENTAL_ESTATE_T VALUES (85, '01/11/2010', 31000, 5000);
INSERT INTO RENTAL_ESTATE_T VALUES (95, '01/12/2010', 35000, 4000);
INSERT INTO RENTAL_ESTATE_T VALUES (105, '01/09/2009', 41000, 4000);
INSERT INTO RENTAL_ESTATE_T VALUES (115, '01/10/2009', 45000, 5000);
INSERT INTO RENTAL_ESTATE_T VALUES (125, '01/05/2010', 31000, 3000);
INSERT INTO RENTAL_ESTATE_T VALUES (135, '01/12/2009', 25500, 4000);
INSERT INTO RENTAL_ESTATE_T VALUES (145, '01/04/2010', 21500, 2000);
INSERT INTO RENTAL_ESTATE_T VALUES (155, '01/01/2010', 25500, 4000);
INSERT INTO RENTAL_ESTATE_T VALUES (165, '01/03/2010', 36000, 3500);
INSERT INTO RENTAL_ESTATE_T VALUES (175, '01/12/2011', 36000, 4500);
INSERT INTO RENTAL_ESTATE_T VALUES (185, '01/01/2011', 37000, 3500);
INSERT INTO RENTAL_ESTATE_T VALUES (195, '01/10/2011', 28500, 2500);
INSERT INTO RENTAL_ESTATE_T VALUES (205, '01/05/2011', 21500, 3000);
INSERT INTO RENTAL_ESTATE_T VALUES (215, '01/03/2011', 35500, 4000);
INSERT INTO RENTAL_ESTATE_T VALUES (225, '01/03/2011', 40000, 4000);
INSERT INTO RENTAL_ESTATE_T VALUES (235, '01/04/2011', 40500, 4000);
INSERT INTO RENTAL_ESTATE_T VALUES (245, '01/05/2011', 28000, 3000);
INSERT INTO RENTAL_ESTATE_T VALUES (255, '01/03/2011', 27000, 2500);
INSERT INTO RENTAL_ESTATE_T VALUES (265, '01/05/2011', 26000, 2500);
INSERT INTO RENTAL_ESTATE_T VALUES (275, '01/01/2011', 25500, 3000);
INSERT INTO RENTAL_ESTATE_T VALUES (285, '01/05/2011', 21500, 3000);
INSERT INTO RENTAL_ESTATE_T VALUES (295, '01/07/2011', 35500, 4000);
```

– جدول العملاء (**CUSTOMER_T**):

إنشاء جدول العملاء

```
CREATE TABLE CUSTOMER_T
        (CUSTOMER_ID        NUMBER       NOT NULL,
        NAME                CHAR(35)     NOT NULL,
        ADDRESS             CHAR(40)     NOT NULL,
        TEL_NO              CHAR(15)     NOT NULL,
        RENTING             CHAR(1)      CHECK (RENTING IN ('T','F')),
        OWNING              CHAR(1)      CHECK (OWNING IN ('T','F')),
        BUYING              CHAR(1)      CHECK (BUYING IN ('T','F')),
   CONSTRAINT CUSTOMER_T_PK PRIMARY KEY (CUSTOMER_ID),
        CHECK ((RENTING IS NOT NULL AND RENTING = 'T ')
            OR (BUYING IS NOT NULL AND BUYING = 'T')
            OR (OWNING IS NOT NULL AND OWNING = 'T')));
```

يتمُّ إدخال البيانات للجدول باستخدام (SQL) حسب التعليمات التالية:

إدخال بيانات العملاء

```
INSERT INTO CUSTOMER_T VALUES
('200', 'Khalid Ahmad Aloufi', 'Al-Rouda Street, Riyadh, SA', '009661-424-2341', 'T', 'T', 'T');
INSERT INTO CUSTOMER_T VALUES
('220', 'Fahad Kahid Alhamid', 'Al-Olya Street, Riyadh, SA', '009661-416-7733', 'T', 'T', 'T');
INSERT INTO CUSTOMER_T VALUES
('240', 'Abdulrahman Mohammed Aleesa', 'King Fahad Street, Jeddah, SA', '009662-444-8932', 'T', 'F', 'T');
INSERT INTO CUSTOMER_T VALUES
('260', 'Khalid Mohammed Alhamad', 'King AbdulAziz Street, Jeddah, SA', '009662-444-5432', 'F', 'F', 'T');
INSERT INTO CUSTOMER_T VALUES
('280', 'Khalid Ahmad Aloufi', 'Al-Hamra Street, Jeddah, SA', '009662-464-2311', 'T', 'T', 'T');
INSERT INTO CUSTOMER_T VALUES
('300', 'Saleh Saeed Alaahrani', 'Al-Reem Street, Dammam, SA', '009663-424-2241', 'T', 'F', 'F');
INSERT INTO CUSTOMER_T VALUES
('320', 'Mohammed Salem Algamdi', 'Al-Nakeel Street, Riyadh, SA', '009661-354-1241', 'F', 'T', 'F');
INSERT INTO CUSTOMER_T VALUES
('340', 'Hmad Abdulaziz Alyousef', 'Al-Salamah Street, Damman, SA', '009663-454-2341', 'T', 'T', 'T');
INSERT INTO CUSTOMER_T VALUES
('360', 'Sultan Hussain Al-Sultan', 'Al-Banat Street, Dammam, SA', '009663-564-5341', 'F', 'T', 'F');
INSERT INTO CUSTOMER_T VALUES
('380', 'Abdulaziz Mohsen Alwabel', 'Hafza Bint Omar Street, Riyadh, SA', '009661-422-2441', 'F', 'T', 'F');
INSERT INTO CUSTOMER_T VALUES
('400', 'Rashed Saleh Alsaleh', 'Al-Moutanabee Street, Riyadh, SA', '009661-454-2341', 'T', 'T', 'T');
INSERT INTO CUSTOMER_T VALUES
('420', 'Abdulkhader Nasser Alhassan', 'Al-Seteen Street, Riyadh, SA', '009661-454-2341', 'T', 'T', 'T');
INSERT INTO CUSTOMER_T VALUES
('440', 'Kheldoon Mohamed Nawaiffa', 'Al-Worood Street, Riyadh, SA', '009661-454-2341', 'T', 'T', 'F');
INSERT INTO CUSTOMER_T VALUES
('460', 'Tareq Salem Alkhalid', 'Al-Zahara Street, Riyadh, SA', '009661-454-2341', 'F', 'T', 'T');
INSERT INTO CUSTOMER_T VALUES
('480', 'Mohsen Ahmad Al-Hamaad', 'Hafsa Bint Ommar Road, Riyadh, SA', '009661-543-9825', 'T', 'T', 'T');
INSERT INTO CUSTOMER_T VALUES
('500', 'Abdulaziz Saleh Al-Zomaan', 'Al-Mutanbi Street, Dammam, SA', '009661-372-7751', 'T', 'T', 'T');
INSERT INTO CUSTOMER_T VALUES
('520', 'Hammad Abdulaziz Al-Hammad', 'Al-Zahara Street, Jeddah, SA', '009661-991-3321', 'F', 'T', 'T');
INSERT INTO CUSTOMER_T VALUES
('540', 'Abdulaziz Tayar Al-Tayar', 'Al-Zahara Street, Jeddah, SA', '009661-441-7561', 'F', 'T', 'T');
```

- جدول عقود الإيجار (**LEASE_T**):

```
إنشاء جدول عقود الإيجار

CREATE TABLE LEASE_T
        (RENTING_CUSTOMER_ID     NUMBER      NOT NULL,
        RENTING_ESTATE_ID        NUMBER      NOT NULL,
        LEASE_DATE               DATE        NOT NULL,
        CONSTRAINT LEASE_T_PK
                PRIMARY KEY (RENTING_CUSTOMER_ID, RENTING_ESTATE_ID),
        CONSTRAINT LEASE_T_FK2 FOREIGN KEY (RENTING_CUSTOMER_ID)
                REFERENCES CUSTOMER_T(CUSTOMER_ID),
        CONSTRAINT LEASE_T_FK3 FOREIGN KEY (RENTING_ESTATE_ID)
                REFERENCES RENTAL_ESTATE_T(RENTAL_ESTATE_ID));
```

يتمُّ إدخال البيانات للجدول باستخدام (SQL) حسب التعليمات التالية:

```
إدخال بيانات عقود الإيجار
INSERT INTO LEASE_T VALUES (240, 55, '01/12/2009');
INSERT INTO LEASE_T VALUES (300, 65, '01/06/2010');
INSERT INTO LEASE_T VALUES (480, 75, '01/09/2009');
INSERT INTO LEASE_T VALUES (420, 85, '01/11/2010');
INSERT INTO LEASE_T VALUES (280, 95, '01/12/2010');
INSERT INTO LEASE_T VALUES (440, 105, '01/09/2009');
```

- جدول ملكية العقارات (**OWNERSHIP_T**):

```
إنشاء جدول ملكية العقارات

CREATE TABLE OWNERSHIP_T
        (OWNER_ID            NUMBER          NOT NULL,
        ESTATE_ID            NUMBER          NOT NULL,
        DEED_ID              CHAR(30)        NOT NULL,
        OWNERSHIP_DATE       DATE            NOT NULL,
        CONSTRAINT OWNERSHIP_T_PK PRIMARY KEY
                        (OWNER_ID, ESTATE_ID),
        CONSTRAINT OWNERSHIP_T_FK1 FOREIGN KEY (OWNER_ID)
                        REFERENCES CUSTOMER_T(CUSTOMER_ID),
        CONSTRAINT OWNERSHIP_T_FK2 FOREIGN KEY (ESTATE_ID)
                        REFERENCES ESTATE_T(ESTATE_ID));
```

يتمُّ إدخال البيانات للجدول باستخدام (SQL) حسب التعليمات التالية:

إدخال بيانات ملكية العقارات

```
INSERT INTO OWNERSHIP_T VALUES (200, 30, 'R1000-2000', '18/03/2000');
INSERT INTO OWNERSHIP_T VALUES (220, 60, 'R2012-2008', '10/07/2008');
INSERT INTO OWNERSHIP_T VALUES (280, 90, 'J0121-2006', '21/11/2006');
INSERT INTO OWNERSHIP_T VALUES (320, 120, 'D1012-2001', '29/04/2001');
INSERT INTO OWNERSHIP_T VALUES (360, 150, 'J1103-2000', '05/02/2005');
INSERT INTO OWNERSHIP_T VALUES (380, 180, 'J1115-2001', '29/04/2003');
INSERT INTO OWNERSHIP_T VALUES (440, 210, 'D1231-2001', '17/11/2001');
INSERT INTO OWNERSHIP_T VALUES (460, 240, 'J1777-1997', '18/01/1997');
INSERT INTO OWNERSHIP_T VALUES (520, 270, 'J1811-1991', '12/07/1991');
INSERT INTO OWNERSHIP_T VALUES (540, 300, 'R1878-2005', '18/09/2006');
INSERT INTO OWNERSHIP_T VALUES (520, 330, 'J2121-2007', '19/10/2007');
INSERT INTO OWNERSHIP_T VALUES (460, 360, 'R0171-2001', '09/01/2004');
INSERT INTO OWNERSHIP_T VALUES (440, 390, 'J0981-2004', '03/02/2002');
INSERT INTO OWNERSHIP_T VALUES (200, 420, 'D1921-2003', '01/05/2003');
INSERT INTO OWNERSHIP_T VALUES (460, 450, 'J1023-2000', '18/05/2009');
INSERT INTO OWNERSHIP_T VALUES (460, 480, 'R1178-1998', '02/07/1998');
INSERT INTO OWNERSHIP_T VALUES (280, 510, 'J1878-2005', '18/09/2005');
INSERT INTO OWNERSHIP_T VALUES (320, 540, 'D1121-2007', '19/10/2007');
INSERT INTO OWNERSHIP_T VALUES (440, 570, 'J0171-2001', '09/01/2001');
INSERT INTO OWNERSHIP_T VALUES (440, 600, 'R0981-2004', '03/02/2004');
INSERT INTO OWNERSHIP_T VALUES (540, 630, 'J1921-2003', '01/05/2003');
INSERT INTO OWNERSHIP_T VALUES (460, 660, 'R1023-2000', '18/05/2000');
```

تابع - إدخال بيانات ملكية العقارات

```
INSERT INTO OWNERSHIP_T VALUES (200, 5, 'R1981-2005', '19/02/2009');
INSERT INTO OWNERSHIP_T VALUES (220, 15, 'R1088-2005', '13/10/2009');
INSERT INTO OWNERSHIP_T VALUES (280, 25, 'R2345-2006', '11/01/2010');
INSERT INTO OWNERSHIP_T VALUES (320, 35, 'R3451-2007', '01/02/2010');
INSERT INTO OWNERSHIP_T VALUES (540, 45, 'J2345-2008', '13/04/2009');
INSERT INTO OWNERSHIP_T VALUES (540, 55, 'J7689-2005', '29/11/2008');
INSERT INTO OWNERSHIP_T VALUES (520, 65, 'J9876-1999', '22/02/2009');
INSERT INTO OWNERSHIP_T VALUES (520, 75, 'D9876-1989', '19/01/2008');
INSERT INTO OWNERSHIP_T VALUES (520, 85, 'D7766-2001', '17/09/2009');
INSERT INTO OWNERSHIP_T VALUES (540, 95, 'J8766-2002', '15/07/2009');
INSERT INTO OWNERSHIP_T VALUES (460, 105, 'R9988-2007', '02/06/2008');
INSERT INTO OWNERSHIP_T VALUES (440, 115, 'R9876-2004', '27/09/2008');
INSERT INTO OWNERSHIP_T VALUES (280, 125, 'J2314-2001', '03/06/2008');
INSERT INTO OWNERSHIP_T VALUES (380, 135, 'J5434-2001', '03/12/2007');
INSERT INTO OWNERSHIP_T VALUES (520, 145, 'J3322-2001', '01/02/2006');
INSERT INTO OWNERSHIP_T VALUES (200, 155, 'R4356-2003', '05/01/2005');
INSERT INTO OWNERSHIP_T VALUES (200, 165, 'R4056-2004', '19/03/2005');
INSERT INTO OWNERSHIP_T VALUES (220, 175, 'R9881-2003', '18/11/2006');
INSERT INTO OWNERSHIP_T VALUES (280, 185, 'J7771-2002', '21/01/2007');
INSERT INTO OWNERSHIP_T VALUES (520, 195, 'D9991-2003', '27/10/2004');
INSERT INTO OWNERSHIP_T VALUES (520, 205, 'J1999-2003', '25/05/2006');
INSERT INTO OWNERSHIP_T VALUES (540, 215, 'R6577-2004', '23/03/2005');
INSERT INTO OWNERSHIP_T VALUES (280, 225, 'D9966-2002', '23/03/2004');
INSERT INTO OWNERSHIP_T VALUES (280, 235, 'J5342-2001', '14/04/2002');
INSERT INTO OWNERSHIP_T VALUES (540, 245, 'R0101-2004', '28/05/2007');
INSERT INTO OWNERSHIP_T VALUES (220, 255, 'J2121-2003', '25/07/2004');
INSERT INTO OWNERSHIP_T VALUES (220, 265, 'R8765-2004', '12/05/2007');
INSERT INTO OWNERSHIP_T VALUES (200, 275, 'D2347-2003', '14/01/2009');
INSERT INTO OWNERSHIP_T VALUES (460, 285, 'R9872-2004', '19/05/2007');
INSERT INTO OWNERSHIP_T VALUES (220, 295, 'J7234-2001', '22/05/2007');
```

- **جدول مبيعات العقارات (SALE_DEAL_T):**

إنشاء جدول مبيعات العقارات

```
CREATE TABLE SALE_DEAL_T
    (BUYING_CUSTOMER_ID    NUMBER     NOT NULL,
    SALE_ESTATE_ID         NUMBER     NOT NULL,
    SALE_PRICE             NUMBER     NOT NULL,
    SALE_DATE              DATE       NOT NULL,
    CONSTRAINT SALE_DEAL_T_PK PRIMARY KEY
      (BUYING_CUSTOMER_ID, SALE_ESTATE_ID),
    CONSTRAINT SALE_DEAL_T_FK1 FOREIGN KEY
      (BUYING_CUSTOMER_ID) REFERENCES CUSTOMER_T(CUSTOMER_ID),
    CONSTRAINT SALE_DEAL_T_FK2 FOREIGN KEY
      (SALE_ESTATE_ID) REFERENCES SALE_ESTATE_T(SALE_ESTATE_ID));
```

ويتمُّ إدخال البيانات للجدول باستخدام (SQL) حسب التعليمات التالية:

إدخال بيانات مبيعات العقارات

إدخال بيانات مبيعات العقارات

```
INSERT INTO SALE_DEAL_T VALUES (240, 30, 766000, '11/05/2000');
INSERT INTO SALE_DEAL_T VALUES (260, 60, 1050000, '16/09/2008');
INSERT INTO SALE_DEAL_T VALUES (240, 90, 500000, '23/11/2007');
INSERT INTO SALE_DEAL_T VALUES (340, 120, 2100000, '01/04/2002');
INSERT INTO SALE_DEAL_T VALUES (400, 150, 1100000, '19/09/2005');
INSERT INTO SALE_DEAL_T VALUES (460, 180, 1500000, '22/04/2004');
INSERT INTO SALE_DEAL_T VALUES (480, 210, 750000, '27/11/2002');
INSERT INTO SALE_DEAL_T VALUES (500, 240, 900000, '07/01/1999');
INSERT INTO SALE_DEAL_T VALUES (540, 270, 4500000, '09/07/1993');
INSERT INTO SALE_DEAL_T VALUES (520, 300, 920000, '22/09/2008');
INSERT INTO SALE_DEAL_T VALUES (500, 330, 970000, '09/10/2008');
INSERT INTO SALE_DEAL_T VALUES (340, 360, 630000, '11/01/2005');
INSERT INTO SALE_DEAL_T VALUES (340, 390, 750000, '14/02/2003');
INSERT INTO SALE_DEAL_T VALUES (240, 420, 955000, '13/05/2003');
INSERT INTO SALE_DEAL_T VALUES (400, 450, 1350000, '25/08/2009');
```

ملحق رقم (2) – 5: استعراض لمحتويات جداول قاعدة البيانات بعد إنشائها في بيئة أوراكل باستخدام تعليمة (SELECT * FROM TableName)

– جدول الموظفين (EMPLOYEE_T):

EMP_ID	FNAME	MNAME	LNAME	POSITION	SALARY	DOB	ADDRESS	TEL_NO
1010	Saleh	Ahmad	Alhamad	Director General	35000	22/05/63	13 Almutanabi Street, Riyadh	009661-454-2341
1020	Abdullah	Abdulrahman	Aloufi	Sale Person	22000	07/10/70	25 Jareer Street, Riyadh	009661-456-7733
1030	Salem	Mohsen	Algamdi	Sale Person	20000	13/09/66	98 Bin Taimiah Street, Jeddah	009662-454-5412
1040	Mishal	Hamad	Alyousef	Sale Person	21500	12/08/69	13 Alsouk Street, Taif	009665-050-3131
1050	Khalid	Saud	Alsultan	Office Manager	30000	22/06/64	22 Bin Handan Street, Jeddah	009665-550-2015
1060	Mohammed	Salman	Abdelaleem	Office Manager	31500	27/08/72	10 Bin Handan Street, Riyadh	009665-555-2345
1070	Sami	Suliman	Aloutaibi	Office Manager	30000	01/11/67	67 Alfadel Street, Damman	009665-050-6178
1080	Saud	Ahmad	Alganim	Office Manager	20000	13/12/72	24 Alfadel Street, AlJouf	009665-550-2134
1090	Abdulrahman	Mansour	Abdulsalam	Sale Person	22000	15/09/75	10 Almadinah Street, Skaka	009665-550-7891
2010	Salman	Mohammed	Alsaleh	Sale Person	22000	26/03/76	15 King Fahad Road, Damman	009665-050-3231
2020	Khalid	Mohammed	Alomar	Sale Person	22000	18/05/61	91 Alwadi Street, Najran	009665-050-1516
2030	Minwer	Hamad	Almutairi	Sale Person	22000	08/11/65	87 Alhamra Road, Jizan	009665-056-1231
2040	Turki	Khalid	Alassaf	Sale Person	22000	15/10/69	25 Prince Abdullah Street, Riyadh	009665-057-8796
2050	Saleh	Hamad	Alzaid	Sale Person	22000	13/07/71	25 King Faisal Street, Riyadh	009665-057-7788
2060	Ghanim	Abdullah	Alhmoud	Sale Person	22000	21/02/69	56 Altahliah Street, Jeddah	009665-055-9781
2070	Sultan	Saleh	Abdulgader	Sale Person	22000	29/04/75	123 Salman Alfarsi Street, Riyadh	009665-059-3831
2080	Suliman	Abdullah	Almushari	Sale Person	22000	05/03/73	45 Prince Sultan Street, Najran	009665-056-3111
2090	Ahmad	Abdullah	Alsaif	Sale Person	22000	18/04/68	13 Khalifa Street, Taif	009665-556-1231
3010	Ahmad	Mansour	Alshemamri	Sale Person	22000	19/09/75	15 Othman street, Jizan	009665-055-3486
3020	Mohammed	Khalid	Alzamil	Sale Person	22000	17/10/61	67 Abubaker Road, Abha	009665-057-6578
3030	Mansour	Abdullah	Alzamil	Office Manager	22000	03/12/60	13 King Abdulaziz Road, Tabouk	009665-054-3331

– جدول الشهادات العلمية (**EMPLOYEE_ACADEMIC_DEGREE_T**):

EMP_ID	DEGREE	INSTITUTE	DATE_ACQ
1010	BA	Louisiana State University	01/05/98
1010	MBA	Louisiana State Universi-ty	12/01/00
1020	BS	University of Pittsburgh	11/03/99
1020	MBA	University of Pittsburgh	17/04/01
1030	BA	King Saud University	13/11/97
1040	BA	King Abdulaziz University	27/02/00
1050	BA	Arkansas State University	12/03/00
1060	BA	University of Oklahoma	10/11/97
1070	BA	King Saud University	27/10/98
1080	BA	King Saud University	27/10/98
1080	MBA	King Saud University	16/12/00
1090	BA	King Abdulaziz University	21/11/96
2010	BA	Philadelphia State Univer-sity	10/08/00
2020	BA	King Abdulaziz University	27/02/00
2030	BA	King Abdulaziz University	27/02/00
2040	BA	King Faisal University	03/02/00
2050	BA	King Faisal University	03/02/00
2060	BA	University of Oklahoma	07/06/00
2070	BA	Philadelphia State Univer-sity	10/08/00
2080	BA	King Saud University	28/11/01
2090	BA	Philadelphia State Univer-sity	10/08/00
3010	BA	King Faisal University	03/02/99
3020	BA	King Faisal University	03/02/99
3030	BA	King Saud University	28/11/01

- جدول مكاتب الشركة (**OFFICE_T**):

```
OFFICE_ID ADDRESS                                                        TEL_NO
--------- -------------------------------------------------------------- ---------------
        1 Head Office, Al-Olya Street, Riyadh, SA                        009661-272-4141
        2 Riyadh Main Office, Al-Farazdaq Street, Al-Malaz, Riyadh, SA   009661-474-3030
        3 Jeddah Main Office, Prince Mohammed Bin Saud Street, Jeddah, SA 009662-272-4141
        4 Dammam Main Office, King Fahad Street, Dammam, SA               009663-231-4141
        5 Jouf Office, Central Street, Al-Jouf, SA                        009664-521-4141
```

– جدول مكاتب عمل الموظفين (**EMPLOYEE_OFFICE_T**):

EMP_ID	OFFICE_ID	STARTING
1010	1	29/04/01
1020	1	14/09/02
1030	3	15/04/02
1040	3	16/02/02
1050	2	29/11/02
1060	3	02/12/02
1070	4	18/10/02
1080	5	13/09/02
1090	5	28/06/02
2010	4	21/06/03
2020	3	20/05/02
2030	3	12/02/03
2040	2	11/08/05
2050	2	17/09/03
2060	3	15/11/04
2070	2	09/12/02
2080	3	07/01/03
2090	4	20/05/07
1070	1	05/12/01
1080	3	09/11/03
1090	3	06/12/01
3010	1	08/06/05
3020	3	06/03/02
3030	5	15/02/01

– جدول العقارات (ESTATE_T):

ESTATE_ID	ADDRESS	E	LISTED_D	OFFICE_ID
30	12 AlMutanbi Street, Riyadh	S	10/05/08	2
60	17 Alolya main Road, Riyadh	S	11/08/08	2
90	12 AlMutanbi Street, Jeddah	S	15/07/07	3
120	17 Alolya main Road, Dammam	S	01/02/02	4
150	12 AlMutanbi Street, Jouf	S	09/07/05	5
180	17 Alolya main Road, Jeddah	S	21/03/04	3
210	12 AlMutanbi Street, Dammam	S	10/12/02	4
240	17 Alolya main Road, Jouf	S	05/11/98	5
270	12 AlMutanbi Street, Jeddah	S	04/03/93	3
300	17 Alolya main Road, Riyadh	S	27/06/08	2
330	12 AlMutanbi Street, Jouf	S	07/02/08	5
360	17 Alolya main Road, Riyadh	S	13/01/04	2
390	12 AlMutanbi Street, Jeddah	S	03/01/03	3
420	17 Alolya main Road, Dammam	S	26/04/03	4
450	12 AlMutanbi Street, Jouf	S	20/02/09	5
480	17 Alolya main Road, Riyadh	S	13/01/09	2
510	12 AlMutanbi Street, Jeddah	S	04/01/08	3
540	17 Alolya main Road, Dammam	S	08/11/07	4
570	12 AlMutanbi Street, Jouf	S	14/08/06	5
600	17 Alolya main Road, Riyadh	S	12/10/08	2
630	12 AlMutanbi Street, Jouf	S	09/11/05	5
660	17 Alolya main Road, Riyadh	S	12/01/07	2
5	10 Salah Eldeen AlAyoubi, Malaz, Riyadh	R	12/04/09	2
15	18 Ahmad bin Hanbal Street, Massem, Riyadh	R	12/01/08	2
25	15 Salah Eldeen AlAyoubi, Malaz, Riyadh	R	12/01/09	2
35	134 Ahmad bin Hanbal Street, Nassen, Riyadh	R	12/01/10	2
45	113 Bin Khldoon Street, Alhamra, Jeddah	R	14/01/09	3
55	18 Alsafa Street, Aljazeera, Jeddah	R	12/06/07	3
65	123 King Abdullaziz Road, Skaka, Jouf	R	22/05/07	5
75	1117 Ammar Bin Yasser Road, Dammam	R	27/10/08	4
85	12 Almutanabi Street, Dammam	R	01/01/08	4
95	11 Prince Syultan Street, Jouf	R	22/12/08	5
105	177 Subah Aljaber, Olya, Riyadh	R	25/01/06	2
115	188 Eygpt Road, Albatha, Riyadh	R	08/09/07	2
125	138 Alyamama Road, Jeddah	R	19/07/08	3
135	183 Khaled Bin Alwaleed, Jeddah	R	16/09/08	3
145	210 Hassan Bin Thabet Street, Jeddah	R	02/03/08	3
155	210 Othman Bin Affan Street, Riyadh	R	14/01/06	2
165	213 King Fahad Road, Riyadh	R	19/05/06	2
175	1010 King Fahad Road, Riyadh	R	14/11/09	2
185	1113 Bin Khldoon Street, Alhamra, Jeddah	R	13/02/07	3
195	1110 Mohammed Bin Nasser Street, Dammam	R	27/11/09	4
205	2313 Sultan Alsaleh Street, Jouf	R	11/01/09	5
215	3210 King Abdullaziz Road, Riyadh	R	17/10/09	2
225	1371 King Fahad Road, Dammam	R	26/03/08	4
235	1020 King Fisal Road, Jeddah	R	07/12/08	3
245	1235 King Khaled Road, Riyadh	R	14/12/07	2
255	128 King Faisl Road, Jouf	R	25/12/06	5
265	143 Salman Alfaresi Street, Riyadh	R	17/09/05	2
275	109 King Abdulaziz Road, Damnam	R	19/10/09	4
285	175 Prince Salman Bin Abdulaziz Road, Riyadh	R	27/08/09	2
295	147 King Fahad Road, Jeddah	R	16/02/08	3

– جدول أسعار العقارات المعروضة للبيع (**SALE_ESTATE_T**):

SALE_ESTATE_ID	ASK_PRICE
30	766000
60	1100000
90	510000
120	2110000
150	1100000
180	1500000
210	750000
240	920000
270	450000
300	930000
330	970000
360	630000
390	750000
420	955000
450	1400000
480	2900000
510	3100000
540	2750000
570	610000
600	4300000
630	830000
660	980000

– جدول أسعار العقارات المعروضة للإيجار (RENTAL_ESTATE_T):

RENTAL_ESTATE_ID	AVAILABI	YEARLY_RENT	INSURANCE_AMOUNT
5	01/05/10	21000	3000
15	01/12/10	25000	4000
25	01/05/11	25000	3000
35	01/04/11	35000	5000
45	01/07/10	31000	5000
55	01/12/09	35000	5000
65	01/05/10	41000	10000
75	01/08/09	35000	5000
85	01/11/10	31000	5000
95	01/12/10	35000	4000
105	01/09/09	41000	4000
115	01/10/09	45000	5000
125	01/05/10	31000	3000
135	01/12/09	25500	4000
145	01/04/10	21500	2000
155	01/01/10	25500	4000
165	01/03/10	36000	3500
175	01/12/11	36000	4500
185	01/01/11	37000	3500
195	01/10/11	28500	2500
205	01/05/11	21500	3000
215	01/03/11	35500	4000
225	01/03/11	40000	4000
235	01/04/11	40500	4000
245	01/05/11	28000	3000
255	01/03/11	27000	2500
265	01/05/11	26000	2500
275	01/01/11	25500	3000
285	01/05/11	21500	3000
295	01/07/11	35500	4000

- جدول العملاء (CUSTOMER_T):

CUSTOMER_ID	NAME	ADDRESS	TEL_NO	R	O	B
200	Khalid Ahmad Aloufi	Al-Rouda Street, Riyadh, SA	009661-424-2341	F	T	F
220	Fahad Kahid Alhamid	Al-Olya Street, Riyadh, SA	009661-416-7733	F	T	F
240	Abdulrahman Mohammed Aleesa	King Fahad Street, Jeddah, SA	009662-444-8932	T	F	T
260	Khalid Mohammed Alhamad	King AbdulAziz Street, Jeddah, SA	009662-444-5432	F	F	T
280	Khalid Ahmad Aloufi	Al-Hamra Street, Jeddah, SA	009662-464-2311	T	T	F
300	Saleh Saeed Alaahrani	Al-Reem Street, Damman, SA	009663-424-2241	T	F	F
320	Mohammed Salem Algamdi	Al-Nakeel Street, Riyadh, SA	009661-354-1241	F	T	F
340	Hmad Abdulaziz Alyousef	Al-Salamah Street, Damman, SA	009663-454-2341	F	F	T
360	Sultan Hussain Al-Sultan	Al-Banat Street, Damman, SA	009663-564-5341	F	T	F
380	Abdulaziz Mohsen Alwabel	Hafza Bint Omar Street, Riyadh, SA	009661-422-2441	F	T	F
400	Rashed Saleh Alsaleh	Al-Houtanabee Street, Riyadh, SA	009661-454-2341	F	F	T
420	Abdulkhader Hasser Alhassan	Al-Seteen Street, Riyadh, SA	009661-454-2341	T	F	F
440	Kheldoon Mohamed Nawaiffa	Al-Vorood Street, Riyadh, SA	009661-454-2341	T	T	F
460	Tareq Salem Alkhalid	Al-Zahara Street, Riyadh, SA	009661-454-2341	F	T	T
480	Mohsen Ahmad Al-Hamaad	Hafsa Bint Ommar Road, Riyadh, SA	009661-543-9825	T	F	T
500	Abdulaziz Saleh Al-Zonaan	Al-Mutanbi Street, Damman, SA	009661-372-7751	F	F	T
520	Hammad Abdulaziz Al-Hammad	Al-Zahara Street, Jeddah, SA	009661-991-3321	F	T	T
540	Abdulaziz Tayar Al-Tayar	Al-Zahara Street, Jeddah, SA	009661-441-7561	F	T	T

– جدول عقود الإيجار (**LEASE_T**):

RENTING_CUSTOMER_ID	RENTING_ESTATE_ID	LEASE_DA
240	55	01/12/09
300	65	01/06/10
480	75	01/09/09
420	85	01/11/10
280	95	01/12/10
440	105	01/09/09

– جدول ملكية العقارات (**OWNERSHIP_T**):

OWNER_ID	ESTATE_ID	DEED_ID	OWNERSHI
200	30	R1000-2000	18/03/00
220	60	R2012-2008	10/07/08
280	90	J0121-2006	21/11/06
320	120	D1012-2001	29/04/01
360	150	J1103-2000	05/02/05
380	180	J1115-2001	29/04/03
440	210	D1231-2001	17/11/01
460	240	J1777-1997	18/01/97
520	270	J1011-1991	12/07/91
540	300	R1878-2005	18/09/06
520	330	J2121-2007	19/10/07
460	360	R0171-2001	09/01/04
440	390	J0981-2004	03/02/02
200	420	D1921-2003	01/05/03
460	450	J1023-2000	18/05/09
460	480	R1178-1998	02/07/98
280	510	J1878-2005	18/09/05
320	540	D1121-2007	19/10/07
440	570	J0171-2001	09/01/01
440	600	R0981-2004	03/02/04
540	630	J1921-2003	01/05/03
460	660	R1023-2000	18/05/00
200	5	R1981-2005	19/02/09
220	15	R1088-2005	13/10/09
280	25	R2345-2006	11/01/10
320	35	R3451-2007	01/02/10
540	45	J2345-2008	13/04/09
540	55	J7689-2005	29/11/08
520	65	J9876-1999	22/02/09
520	75	D9876-1989	19/01/08
520	85	D7766-2001	17/09/09
540	95	J8766-2002	15/07/09
460	105	R9988-2007	02/06/08
440	115	R9876-2004	27/09/08
280	125	J2314-2001	03/06/08
380	135	J5434-2001	03/12/07
520	145	J3322-2001	01/02/06
200	155	R4356-2003	05/01/05
200	165	R4056-2004	19/03/05
220	175	R9881-2003	18/11/06
280	185	J7771-2002	21/01/07
520	195	D9991-2003	27/10/04
520	205	J1999-2003	25/05/06
540	215	R6577-2004	23/03/05
280	225	D9966-2002	23/03/04
280	235	J5342-2001	14/04/02
540	245	R0101-2004	28/05/07
220	255	J2121-2003	25/07/04
220	265	R8765-2004	12/05/07
200	275	D2347-2003	14/01/09
460	285	R9872-2004	19/05/07
220	295	J7234-2001	22/05/07

– جدول مبيعات العقارات (**SALE_DEAL_T**):

BUYING_CUSTOMER_ID	SALE_ESTATE_ID	SALE_PRICE	SALE_DAT
240	30	766000	11/05/00
260	60	1050000	16/09/08
240	90	500000	23/11/07
340	120	2100000	01/04/02
400	150	1100000	19/09/05
460	180	1500000	22/04/04
480	210	750000	27/11/02
500	240	900000	07/01/99
540	270	4500000	09/07/93
520	300	920000	22/09/08
500	330	970000	09/10/08
340	360	630000	11/01/05
340	390	750000	14/02/03
240	420	955000	13/05/03
400	450	1350000	25/08/09

ملحق رقم (2) – 6: تمارين تطبيقية على لغة الاستفسار البنائية (SQL):

1- ما أرقام وأسماء ووظائف الموظفين الذين يسكنون في منطقة الرياض 'Riyadh'؟

الحل:

```
SELECT Emp_Id, FName, MName, LName, Position
FROM EMPLOYEE_T
WHERE Address LIKE '%Riyadh%';
```

النتيجة:

EMP_ID	FNAME	MNAME	LNAME	POSITION
1010	Saleh	Ahmad	Alhamad	Director General
1020	Abdullah	Abdulrahman	Aloufi	Sale Person
1060	Mohammed	Salman	Abdelaleem	Office Manager
2040	Turki	Khalid	Alassaf	Sale Person
2050	Saleh	Hamad	Alzaid	Sale Person
2070	Sultan	Saleh	Abdulgader	Sale Person

2- ما أرقام وأسماء ووظائف الموظفين الذين يسكنون في منطقة جدة 'Jeddah'، مرتبة تنازلياً حسب أسماء عائلاتهم؟

الحل:

```
SELECT Emp_Id, FName, MName, LName, Position
FROM EMPLOYEE_T
WHERE Address LIKE '%Jeddah%'
ORDER BY LName DESC;
```

النتيجة:

EMP_ID	FNAME	MNAME	LNAME	POSITION
1050	Khalid	Saud	Alsultan	Office Manager
2060	Ghanim	Abdullah	Alhmoud	Sale Person
1030	Salem	Mohsen	Algamdi	Sale Person

3- أنشئ منظوراً بمُسمَّى Employee_V يحتوي على أرقام الموظفين الذين لا يعملون في مدينة الرياض ومدينة جدة، وأسمائهم، ووظائفهم، ورواتبهم، وعناوينهم البريدية. استعرض بيان المنظور؛ للتأكُّد من البيانات التي يحتويها.

الحل:

```
CREATE VIEW EMPLOYEE_V AS
SELECT Emp_ID, FName, MName, LName, Position, Salary, Address
FROM EMPLOYEE_T
WHERE ADDRESS NOT LIKE '%Riyadh%' AND ADDRESS NOT LIKE '%Jeddah%';
```

النتيجة:

EMP_ID	FNAME	MNAME	LNAME	POSITION	SALARY	ADDRESS
1040	Mishal	Hamad	Alyousef	Sale Person	21500	13 Alsouk Street, Taif
1070	Sami	Suliman	Aloutaibi	Office Manager	30000	67 Alfadel Street, Damman
1080	Saud	Ahmad	Alganim	Office Manager	20000	24 Alfadel Street, AlJouf
1090	Abdulrahman	Mansour	Abdulsalam	Sale Person	22000	10 Alnadinah Street, Skaka
2010	Salman	Mohammed	Alsaleh	Sale Person	22000	15 King Fahad Road, Dammam
2020	Khalid	Mohammed	Alonar	Sale Person	22000	91 Aluadi Street, Najran
2030	Minwer	Hamad	Almutairi	Sale Person	22000	87 Alhamra Road, Jizan
2080	Suliman	Abdullah	Almushari	Sale Person	22000	45 Prince Sultan Street, Najran
2090	Ahmad	Abdullah	Alsaif	Sale Person	22000	13 Khalifa Street, Taif
3010	Ahmad	Mansour	Alshemanri	Sale Person	22000	15 Othman street, Jizan
3020	Mohammed	Khalid	Alzamil	Sale Person	22000	67 Abubaker Road, Abha
3030	Mansour	Abdullah	Alzamil	Office Manager	22000	13 King Abdulaziz Road, Tabouk

4- باستخدام المنظور الذي قمت بإنشائه، أظهر أعلى راتب، وأقل راتب، ومتوسط رواتب الموظفين الذين لا يعملون في مدينة الرياض ومدينة جدة.

الحل:

```
SELECT MAX(Salary), MIN(Salary), AVG(Salary)
FROM EMPLOYEE_V;
```

النتيجة:

```
MAX(SALARY) MIN(SALARY) AVG(SALARY)
----------- ----------- -----------
      30000       20000  22458.3333
```

5- ما رقم وعنوان المكتب الذي يعمل فيه الموظف Salem Mohsen Algamdi؟

الحل:

```
SELECT Office_ID, Address
FROM (EMPLOYEE_T NATURAL JOIN EMPLOYEE_OFFICE_T) JOIN OFFICE_T
On EMPLOYEE_OFFICE_T.Office_ID = OFFICE_T.Office_ID
WHERE FName='Salem' and MName='Mohsen' and LName='Algamdi';
```

النتيجة:

```
OFFICE_ID ADDRESS
--------- ---------------------------------------------------------------
        3 Jeddah Main Office, Prince Mohammed Bin Saud Street, Jeddah, SA
```

6- ما أرقام وعناوين وإيجارات العقارات المعروضة للإيجار من خلال المكتب رقم (2) ويقل إيجارها عن 000,30؟

الحل:

```
SELECT Estate_Id, Address, Yearly_Rent
FROM ESTATE_T Join RENTAL_ESTATE_T
ON ESTATE_T.Estate_ID = RENTAL_ESTATE_T.Rental_Estate_ID
WHERE Yearly_Rent < 30000 and ESTATE_T.Office_ID =2;
```

النتيجة:

ESTATE_ID	ADDRESS	YEARLY_RENT
5	13 Salah Eldeen AlAyoubi, Malaz, Riyadh	21000
15	18 Ahmad bin Hanbal Street, Nassem, Riyadh	25000
25	15 Salah Eldeen AlAyoubi, Malaz, Riyadh	25000
155	218 Othman Bin Affan Street, Riyadh	25500
245	1235 King Khaled Road, Riyadh	28000
265	143 Salman Alfaresi Street, Riyadh	26000
285	175 Prince Salman Bin Abdulaziz Road, Riyadh	21500

7- ما أسماء ورواتب الموظفين الذين تنحصر رواتبهم بين 000,15 و 000,20 مرتبة تصاعدياً حسب الراتب؟

الحل:

```
SELECT FName, MName, LName, Salary
FROM EMPLOYEE_T
WHERE Salary BETWEEN 15000 AND 20000
ORDER BY Salary ASC;
```

النتيجة:

```
FNAME        MNAME        LNAME        SALARY
------------ ------------ ------------ ----------
Saud         Ahmad        Alganim      20000
Salem        Mohsen       Alqamdi      20000
```

8- ما أسماء ورواتب الموظفين الذين رواتبهم 000,20 أو 000,30 مرتبة تصاعدياً حسب الراتب وتنازلياً حسب اسم العائلة (بداخل الراتب)؟

الحل:

```
SELECT FName, MName, LName, Salary
FROM EMPLOYEE_T
WHERE SALARY IN (20000, 30000)
ORDER BY Salary ASC, LName DESC;
```

النتيجة:

```
FNAME        MNAME        LNAME        SALARY
------------ ------------ ------------ ----------
Saud         Ahmad        Alganim      20000
Salem        Mohsen       Algamdi      20000
Khalid       Saud         Alsultan     30000
Sami         Suliman      Aloutaibi    30000
```

9- مَنْ هم العملاء المشترون والمستأجرون في نفس الوقت من الشركة؟

الحل:

```
SELECT *
FROM CUSTOMER_T
WHERE Renting='T' AND Buying='T';
```

النتيجة:

CUSTOMER_ID	NAME	ADDRESS	TEL_NO	R	O	B
240	Abdulrahman Mohammed Aleesa	King Fahad Street, Jeddah, SA	009662-444-8932	T	F	T
400	Mohsen Ahmad Al-Hamaad	Hafsa Bint Omnar Road, Riyadh, SA	009661-543-9825	T	F	T

10- ما أرقام العقارات المعروضة للإيجار والجاهزة للسكن بتاريخ 1/3/2011 أو بعد هذا التاريخ، ويتراوح إيجارها السنوي بين 000,25 و 000,30 مرتبة تصاعدياً حسب تاريخ جاهزيتها للسكن؟

الحل:

```
SELECT Rental_Estate_ID, Yearly_Rent, Availability_Date
FROM RENTAL_ESTATE_T
WHERE Availability_Date >= '01-03-2011' AND
(Yearly_Rent BETWEEN 25000 AND 30000)
ORDER BY Availability_Date;
```

النتيجة:

RENTAL_ESTATE_ID	YEARLY_RENT	AVAILABI
255	27000	01/03/11
265	26000	01/05/11
25	25000	01/05/11
245	28000	01/05/11
195	28500	01/10/11

11- ما رقم كلِّ مكتب من مكاتب الشركة يزيد فيه عدد الموظفين عن 3 موظفين، وما عدد الموظفين العاملين فيه؟ أظهر عدد العاملين في هذه المكاتب بمسمى 'No. Of Emp'.

الحل:

```
SELECT Office_ID, COUNT(*) "No. Of Emp"
FROM EMPLOYEE_OFFICE_T
GROUP BY Office_ID
HAVING COUNT(*) > 3;
```

النتيجة:

OFFICE_ID	No. Of Emp
1	4
2	4
3	10

12- ما رقم كل موظف من موظفي الشركة وعدد الشهادات الحاصل عليها؟ أظهر عدد الشهادات تحت مُسمَّى 'No. Of Degrees'، ورتب النتيجة تصاعدياً حسب رقم الموظف.

الحل:

```
SELECT Emp_ID, COUNT(*) "No. Of Degrees"
FROM EMPLOYEE_ACADEMIC_DEGREE_T
GROUP BY Emp_ID
ORDER BY Emp_ID;
```

النتيجة:

EMP_ID	No. Of Degrees
1010	2
1020	2
1030	1
1040	1
1050	1
1060	1
1070	1
1080	2
1090	1
2010	1
2020	1
2030	1
2040	1
2050	1
2060	1
2070	1
2080	1
2090	1
3010	1
3020	1
3030	1

13- ما أرقام وأسماء وعدد شهادات الموظفين الحاصلين على أكثر من شهادة علمية واحدة؟

الحل:

```
SELECT Emp_ID, FName, MName, LName, COUNT(*)
FROM EMPLOYEE_T NATURAL JOIN EMPLOYEE_ACADEMIC_DEGREE_T
GROUP BY Emp_ID, FName, MName, LName
HAVING COUNT(*) >1;
```

النتيجة:

```
EMP_ID FNAME        MNAME        LNAME          COUNT(*)
------ ------------ ------------ ------------ ----------
  1080 Saud         Ahmad        Alganim               2
  1010 Saleh        Ahmad        Alhamad               2
  1020 Abdullah     Abdulrahman  Aloufi                2
```

14- ما متوسط أسعار بيع العقارات في العام 2008؟ أظهِر النتيجة تحت مُسمَّى 'Average Sale PriCe In 2008'.

الحل:

```
SELECT AVG(Sale_Price) "Average Sale Price In 1988"
FROM SALE_DEAL_T
WHERE Sale_Date LIKE '%08';
```

أو

```
SELECT AVG(Sale_Price) "Average Sale Price In 1988"
FROM SALE_DEAL_T
WHERE Sale_Date BETWEEN '01/01/2008' AND '31/12/2008';
```

النتيجة:

```
Average Sale Price In 1988
--------------------------
                    980000
```

15- ما مجموع أرباح الشركة الفعلية من بيع العقارات إذا علمت أن الشركة تأخذ ما نسبته 5.2% من قيمة بيع العقار الواحد أرباحاً لها عن كل عملية بيع؟ أظهر مجموع الأرباح تحت مُسمى 'Total Sale Profit'؟

الحل:

```
SELECT SUM(Sale_Price) * 0.025 "Total Sale Profit"
FROM SALE_DEAL_T;
```

النتيجة:

```
Total Sale Profit
-----------------
           468525
```

16- ما أرقام وأسماء ورواتب وعدد شهادات كل موظف من الموظفين الذين يعملون في المكتب رقم (2)؟

الحل:

```
SELECT Emp_ID, FName, MName, LName, Salary, COUNT(*)
FROM (EMPLOYEE_T JOIN EMPLOYEE_ACADEMIC_DEGREE_T
ON EMPLOYEE_T.Emp_ID=EMPLOYEE_ACADEMIC_DEGREE_T.Emp_ID)
JOIN EMPLOYEE_OFFICE_T
ON EMPLOYEE_T.Emp_ID=EMPLOYEE_OFFICE_T.Emp_ID
WHERE Office_ID = 2
GROUP BY Emp_ID, FName, MName, LName, Salary;
```

النتيجة:

EMP_ID	FNAME	MNAME	LNAME	SALARY	COUNT(*)
2050	Saleh	Hamad	Alzaid	22000	1
2040	Turki	Khalid	Alassaf	22000	1
1050	Khalid	Saud	Alsultan	30000	1
2070	Sultan	Saleh	Abdulgader	22000	1

17- ما أسماء الموظفين الذين عملوا في أكثر من مكتب واحد من مكاتب الشركة وعدد المكاتب التي عملوا فيها؟

الحل:

```
SELECT FName, MName, LName, COUNT(*)
FROM EMPLOYEE_T NATURAL JOIN EMPLOYEE_OFFICE_T
GROUP BY FName, MName, LName
HAVING COUNT(*) > 1;
```

النتيجة:

FNAME	MNAME	LNAME	COUNT(*)
Saud	Ahmad	Alganim	2
Sami	Suliman	Aloutaibi	2
Abdulrahman	Mansour	Abdulsalam	2

18- ما أرقام وعناوين المكاتب وعدد العقارات المعروضة في كل منها سواءً أكانت العقارات للبيع أو للإيجار، مرتبة تصاعدياً حسب رقم المكتب؟ أظهر عدد العقارات تحت مسمى «No. Of Estates».

الحل:

```
SELECT OFFICE_T.Office_ID, OFFICE_T.Address, COUNT(*) "No. Of Estates"
FROM ESTATE_T JOIN OFFICE_T ON ESTATE_T.Office_ID=OFFICE_T.Office_ID
GROUP BY OFFICE_T.Office_ID, OFFICE_T.Address
ORDER BY OFFICE_T.Office_ID;
```

النتيجة:

OFFICE_ID	ADDRESS	No. Of Estates
2	Riyadh Main Office, Al-Farazdaq Street, Al-Malaz, Riyadh, SA	20
3	Jeddah Main Office, Prince Mohammed Bin Saud Street, Jeddah, SA	13
4	Dammam Main Office, King Fahad Street, Damman, SA	9
5	Jouf Office, Central Street, Al-Jouf, SA	10

19- ما أرقام وأسماء العملاء الذين تم بيع عقارات لهم، وما عدد وإجمالي بيع العقارات التي بيعت لكل واحد منهم؟

الحل:

```
SELECT Owner_ID, Name, COUNT(*), SUM(Sale_Price)
FROM (OWNERSHIP_T JOIN SALE_DEAL_T
ON OWNERSHIP_T.Estate_ID=SALE_DEAL_T.Sale_Estate_ID)
JOIN CUSTOMER_T ON Customer_ID=Owner_ID
GROUP BY Owner_ID, Name;
```

النتيجة:

OWNER_ID	NAME	COUNT(*)	SUM(SALE_PRICE)
320	Mohammed Salem Algamdi	1	2100000
360	Sultan Hussain Al-Sultan	1	1100000
520	Hammad Abdulaziz Al-Hammad	2	5470000
220	Fahad Kahid Alhamid	1	1050000
280	Khalid Ahmad Aloufi	1	500000
200	Khalid Ahmad Aloufi	2	1721000
380	Abdulaziz Mohsen Alwabel	1	1500000
440	Kheldoon Mohamed Nawaiffa	2	1500000
540	Abdulaziz Tayar Al-Tayar	1	920000
460	Tareq Salem Alkhalid	3	2880000

20- ما أرقام وحجم مبيعات كل مكتب من مكاتب الشركة قام ببيع عقارات تزيد قيمتها عن 4,000,000؟

الحل:

```
SELECT Office_ID, SUM(Sale_Price)
FROM ESTATE_T JOIN SALE_DEAL_T ON Estate_ID=Sale_Estate_ID
GROUP BY Office_ID
HAVING SUM(Sale_Price) > 4000000;
```

النتيجة:

OFFICE_ID	SUM(SALE_PRICE)
5	4320000
3	7250000

21- ما أسماء الموظفين الحاصلين على أكثر من مؤهل علمي واحد؟

الحل:

```
SELECT Emp_ID, FName, MName, LName
FROM EMPLOYEE_T
WHERE Emp_ID IN (SELECT Emp_ID
FROM EMPLOYEE_T NATURAL JOIN EMPLOYEE_ACADEMIC_DEGREE_T
GROUP BY Emp_ID
HAVING COUNT(*) > 1);
```

النتيجة:

EMP_ID	FNAME	MNAME	LNAME
1010	Saleh	Ahmad	Alhamad
1020	Abdullah	Abdulrahman	Aloufi
1080	Saud	Ahmad	Alganim

22- ما إجمالي عدد الموظفين (سواءً الذين عملوا أو ما زالوا يعملون) في كل مكتب من مكاتب الشركة؟

الحل:

```
SELECT Office_ID, COUNT(*)
FROM (EMPLOYEE_T E JOIN EMPLOYEE_OFFICE_T EO ON
E.Emp_ID=EO.Emp_ID) JOIN OFFICE_T O ON EO.Office_ID=O.Office_ID
GROUP BY Office_ID;
```

النتيجة:

```
OFFICE_ID   COUNT(*)
---------- ----------
         1          4
         2          4
         4          3
         5          3
         3         10
```

23- ما أرقام وعناوين العقارات التي بيعت (Sale_PriCe) بسعر أقل من القيمة التي عرضت فيها (Ask_PriCe)؟

الحل:

-

```
SELECT Estate_ID, Address
FROM (ESTATE_T JOIN SALE_ESTATE_T ON Estate_ID=Sale_Estate_ID)
JOIN SALE_DEAL_T ON ESTATE_T.Estate_ID=SALE_DEAL_T.Sale_Estate_ID
WHERE Sale_Price < Ask_Price;
```

النتيجة:

```
ESTATE_ID ADDRESS
--------- ---------------------------
       60 17 Alolya main Road, Riyadh
       90 12 AlMutanbi Street, Jeddah
      120 17 Alolya main Road, Dammam
      240 17 Alolya main Road, Jouf
      300 17 Alolya main Road, Riyadh
      450 12 AlMutanbi Street, Jouf
```

24- ما رقم مكتب الشركة الذي يعمل فيه كل موظف حالياً (آخر مكتب وجه للعمل فيه)، وراتبه الحالي وعدد شهاداته؟ (ملاحظة: قد يعمل الموظف في أكثر من مكتب ولكن في فترات زمنية مختلفة. المطلوب الأخذ بعين الاعتبار آخر مكتب يعمل فيه الموظف (وهو مقر عمله الحالي)).

الحل:

```
SELECT E1.Emp_ID, E1.FName, E1.MName, E1.LName, Office_ID, Salary, Starting_Date,
COUNT(Degree) No_Of_Degrees
FROM ((EMPLOYEE_T E1 JOIN EMPLOYEE_OFFICE_T EO1 ON E1.Emp_ID = EO1.Emp_ID)
JOIN OFFICE_T O1 ON EO1.Office_ID = O1.Office_ID)
JOIN EMPLOYEE_ACADEMIC_DEGREE_T EAD1 ON E1.Emp_ID = EAD1.Emp_ID
WHERE Starting_Date =
(SELECT MAX(Starting_Date)
FROM EMPLOYEE_T E2 JOIN EMPLOYEE_OFFICE_T EO2 ON E2.Emp_ID = EO2.Emp_ID
WHERE E1.Emp_ID = E2.Emp_ID)
GROUP BY E1.Emp_ID, E1.FName, E1.MName, E1.LName, Office_ID, Salary, Starting_Date;
```

النتيجة:

EMP_ID	FNAME	MNAME	LNAME	OFFICE_ID	SALARY	STARTING	NO_OF_DEGREES
3020	Mohammed	Khalid	Alzamil	3	22000	06/03/02	1
1030	Salem	Mohsen	Algamdi	3	20000	15/04/02	1
1050	Khalid	Saud	Alsultan	2	30000	29/11/02	1
2030	Minwer	Hamad	Almutairi	3	22000	12/02/03	1
2040	Turki	Khalid	Alassaf	2	22000	11/08/05	1
2050	Saleh	Hamad	Alzaid	2	22000	17/09/03	1
2080	Suliman	Abdullah	Almushari	3	22000	07/01/03	1
3030	Mansour	Abdullah	Alzamil	5	22000	15/02/01	1
1010	Saleh	Ahmad	Alhamad	1	35000	29/04/01	2
2010	Salman	Mohammed	Alsaleh	4	22000	21/06/03	1
2090	Ahmad	Abdullah	Alsaif	4	22000	20/05/07	1
1040	Mishal	Hamad	Alyousef	3	21500	16/02/02	1
1070	Sami	Suliman	Aloutaibi	4	30000	18/10/02	1
2020	Khalid	Mohammed	Alomar	3	22000	20/05/02	1
1080	Saud	Ahmad	Alganin	3	20000	09/11/03	2
1020	Abdullah	Abdulrahman	Aloufi	1	22000	14/09/02	2
1090	Abdulrahman	Mansour	Abdulsalam	5	22000	28/06/02	1
2060	Ghanim	Abdullah	Alhmoud	3	22000	15/11/04	1
1060	Mohammed	Salman	Abdelaleem	3	31500	02/12/02	1
2070	Sultan	Saleh	Abdulgader	2	22000	09/12/02	1
3010	Ahmad	Mansour	Alshemanri	1	22000	08/06/05	1

25- ما هو العدد الأكبر من الشهادات العلمية للموظفين في كل مكتب، وما هو أعلى مرتب يتقاضاه الموظفون في نفس المكتب؟ (ملاحظة: قد يعمل الموظف في أكثر من مكتب ولكن في فترات زمنية مختلفة. المطلوب الأخذ بعين الاعتبار آخر مكتب يعمل فيه الموظف (وهو مقر عمله الحالي)).

الحل:

```
SELECT Office_ID Office, MAX(Salary) MAX_Office_Salary, MAX(No_Of_Degrees)
Max_Office_Degrees
FROM (SELECT E1.Emp_ID, E1.FName, E1.MName, E1.LName, Office_ID, Salary,
Starting_Date, COUNT(Degree) No_Of_Degrees
FROM ((EMPLOYEE_T E1 JOIN EMPLOYEE_OFFICE_T EO1 ON
E1.Emp_ID = EO1.Emp_ID)
JOIN OFFICE_T O1 ON EO1.Office_ID = O1.Office_ID)
JOIN EMPLOYEE_ACADEMIC_DEGREE_T EAD1 ON E1.Emp_ID = EAD1.Emp_ID
WHERE Starting_Date =
(SELECT MAX(Starting_Date)
FROM EMPLOYEE_T E2 JOIN EMPLOYEE_OFFICE_T EO2 ON E2.Emp_ID =
EO2.Emp_ID
WHERE E1.Emp_ID = E2.Emp_ID)
GROUP BY E1.Emp_ID, E1.FName, E1.MName, E1.LName, Office_ID, Salary,
Starting_Date)
GROUP BY Office_ID;
```

النتيجة:

OFFICE	MAX_OFFICE_SALARY	MAX_OFFICE_DEGREES
1	35000	2
2	30000	1
4	30000	1
5	22000	1
3	31500	2

26- ما رقم المكتب الذي يتقاضى موظفوه أعلى متوسط للرواتب (بغض النظر عن المكتب الذي يعمل فيه كلٌّ موظف سواءً حالياً أو فيما سبق)، وما هو هذا المتوسط؟

الحل:

```
SELECT OFFICE_T.Office_ID, AVG(Salary) Maximum_Average_Salary
FROM (OFFICE_T JOIN EMPLOYEE_OFFICE_T
ON OFFICE_T.Office_ID = EMPLOYEE_OFFICE_T.Office_ID) JOIN EMPLOYEE_T
ON EMPLOYEE_OFFICE_T.Emp_ID = EMPLOYEE_T.Emp_ID
GROUP BY OFFICE_T.Office_ID
HAVING AVG(Salary) =
(SELECT MAX(Average_Salary)
FROM (SELECT AVG(Salary) Average_Salary
FROM (OFFICE_T JOIN EMPLOYEE_OFFICE_T
ON OFFICE_T.Office_ID = EMPLOYEE_OFFICE_T.Office_ID) JOIN EMPLOYEE_T
ON EMPLOYEE_OFFICE_T.Emp_ID = EMPLOYEE_T.Emp_ID
GROUP BY OFFICE_T.Office_ID));
```

النتيجة:

```
OFFICE_ID MAXIMUM_AVERAGE_SALARY
--------- ----------------------
        1                  27250
```

27- ما اسم العميل (أو العملاء) الذين يملكون أكبر عددٍ من العقارات المعروضة من قبل الشركة (سواء للإيجار أو للبيع)، وما هو عدد هذه العقارات؟

الحل:

```
SELECT Owner_ID, Name, COUNT(*) MAX_Owned_Estates
FROM (CUSTOMER_T JOIN OWNERSHIP_T ON Customer_ID=Owner_ID)
GROUP BY Owner_ID, Name
HAVING COUNT(*) =
(SELECT MAX(No_Of_Owned_Estates) MAX_No_Of_Owned_Estates
FROM (SELECT Owner_ID, COUNT(*) No_Of_Owned_Estates
FROM OWNERSHIP_T
GROUP BY Owner_ID));
```

النتيجة:

```
OWNER_ID NAME                                MAX_OWNED_ESTATES
--------- ---------------------------------- -----------------
      520 Hammad Abdulaziz Al-Hammad                         8
```

28- ما أقدم عقار معروض للبيع (من حيث تاريخ عرضه)، ولم يتم بيعه حتى الآن؟

الحل:

```
SELECT Estate_ID, Address, Listed_Date
FROM ESTATE_T
WHERE (Estate_Type = 'S') AND Listed_Date=(SELECT MIN(Listed_Date)
FROM ESTATE_T
WHERE Estate_ID IN
(SELECT Sale_Estate_ID
FROM SALE_ESTATE_T
MINUS
SELECT Sale_Estate_ID
FROM SALE_DEAL_T));
```

النتيجة:

```
ESTATE_ID ADDRESS                                           LISTED_D
---------- ------------------------------------------------- --------
       630 12 AlMutanbi Street, Jouf                         09/11/05
```

29- ما أقدم عقار معروض سواء للبيع أو للإيجار (من حيث تاريخ عرضه) ولم يتم بيعه أو إيجاره حتى الآن؟

الحل:

```
SELECT Estate_ID, Address, Listed_Date
FROM ESTATE_T
WHERE Listed_Date=(SELECT MIN(Listed_Date)
FROM ESTATE_T
WHERE Estate_ID IN
((SELECT Rental_Estate_ID
FROM RENTAL_ESTATE_T
MINUS
SELECT Renting_Estate_ID
FROM LEASE_T)
UNION
(SELECT Sale_Estate_ID
FROM SALE_ESTATE_T
MINUS
SELECT Sale_Estate_ID
FROM SALE_DEAL_T)));
```

النتيجة:

```
ESTATE_ID ADDRESS                                             LISTED_D
--------- --------------------------------------------------- --------
      265 143 Salman Alfaresi Street, Riyadh                  17/09/05
```

30- ما عدد العقارات المعروضة (أو التي عرضت) سواء للبيع أو للإيجار في كل مكتب من مكاتب الشركة؟

الحل:

```
SELECT OFFICE_T.Office_ID, COUNT(*)
FROM OFFICE_T JOIN ESTATE_T ON
OFFICE_T.Office_ID=ESTATE_T.Office_ID
GROUP BY OFFICE_T.Office_ID;
```

النتيجة:

```
OFFICE_ID   COUNT(*)
---------- ----------
         2         20
         4          9
         5         10
         3         13
```

31- ما عدد العقارات المعروضة سواء للبيع أو للإيجار في كل مكتب من مكاتب الشركة خلاف تلك التي تمَّ بيعها أو إيجارها من قِبل المكتب؟

الحل:

```
SELECT OFFICE_T.Office_ID, OFFICE_T.Address, COUNT(*)
FROM OFFICE_T JOIN ESTATE_T ON
OFFICE_T.Office_ID = ESTATE_T.Office_ID
WHERE ESTATE_T.Estate_ID NOT IN
(SELECT Sale_Estate_ID
FROM SALE_DEAL_T)
AND
ESTATE_T.Estate_ID NOT IN
(SELECT Renting_Estate_ID
FROM LEASE_T)
GROUP BY OFFICE_T.Office_ID, OFFICE_T.Address;
```

النتيجة:

```
OFFICE_ID ADDRESS                                                            COUNT(*)
---------- ------------------------------------------------------------------ ----------
         3 Jeddah Main Office, Prince Mohammed Bin Saud Street, Jeddah, SA            8
         5 Jouf Office, Central Street, Al-Jouf, SA                                   4
         4 Dammam Main Office, King Fahad Street, Dammam, SA                          4
         2 Riyadh Main Office, Al-Farazdaq Street, Al-Malaz, Riyadh, SA              15
```

32- ما اسم الموظف ورقم المكتب الذي يعمل فيه حالياً ويتقاضى راتباً يزيد عن أحد الموظفين الذين يعملون معه في نفس المكتب على الرغم من أنه يحمل شهادات علمية أقل؟

الحل:

```
SELECT Emp_ID, FName, MName, LName, Salary, No_Of_Degrees, Office_ID
FROM (SELECT E1.Emp_ID, E1.FName, E1.MName, E1.LName, Salary,
COUNT(DEGREE) No_Of_Degrees, Office_ID
FROM ((EMPLOYEE_T E1 JOIN EMPLOYEE_OFFICE_T EO1 ON
E1.Emp_ID=EO1.Emp_ID)
JOIN OFFICE_T O1 ON EO1.Office_ID=O1.Office_ID)
JOIN EMPLOYEE_ACADEMIC_DEGREE_T EAD1 ON E1.Emp_ID=EAD1.Emp_ID
WHERE Starting_Date=(SELECT MAX(Starting_Date)
FROM EMPLOYEE_T E2 JOIN EMPLOYEE_OFFICE_T EO2 ON
E2.Emp_ID=EO2.Emp_ID
WHERE E1.Emp_ID=E2.Emp_ID)
GROUP BY E1.Emp_ID, E1.FName, E1.MName, E1.LName, Salary, Office_ID) TEMP
WHERE
Salary > ANY (SELECT Salary
FROM (SELECT E2.Emp_ID, E2.FName, E2.MName, E2.LName, Salary,
COUNT(Degree) No_Of_Degrees, Office_ID
FROM ((EMPLOYEE_T E2 JOIN EMPLOYEE_OFFICE_T EO2 ON
E2.Emp_ID=EO2.Emp_ID)
JOIN OFFICE_T O2 ON EO2.Office_ID=O2.Office_ID)
JOIN EMPLOYEE_ACADEMIC_DEGREE_T EAD2 ON E2.Emp_ID=EAD2.EMP_ID
WHERE Starting_Date=(SELECT MAX(Starting_Date)
FROM EMPLOYEE_T E3 JOIN EMPLOYEE_OFFICE_T EO3
ON E3.Emp_ID=EO3.Emp_ID
WHERE E2.Emp_ID = E3.Emp_ID)
GROUP BY E2.Emp_ID, E2.FName, E2.MName, E2.LName, Salary, Office_ID)
WHERE TEMP.Emp_ID <> Emp_ID
AND TEMP.Office_ID = Office_ID
AND TEMP.No_Of_Degrees < No_Of_Degrees);
```

النتيجة:

EMP_ID	FNAME	MNAME	LNAME	SALARY	NO_OF_DEGREES	OFFICE_ID
2080	Suliman	Abdullah	Almushari	22000	1	3
2020	Khalid	Mohammed	Alomar	22000	1	3
1040	Mishal	Hamad	Alyousef	21500	1	3
2030	Minwer	Hamad	Almutairi	22000	1	3
2060	Ghanim	Abdullah	Alhmoud	22000	1	3
3020	Mohammed	Khalid	Alzamil	22000	1	3
1060	Mohammed	Salman	Abdelaleem	31500	1	3

ملحق رقم (3)

ترجمة المصطلحات

الترجمة	المصطلح
عبارة الانسحاب (أو التراجع)	Abort (or RollbaCk) Statement
فعل	ACtion
التصويرة اللاحقة	After Image
دوال التجميع	Aggregate FunCtions
تجميع	Aggregation
الاسم المستعار	Alias
مرحلة التحليل	Analysis Phase
نظام تطبيقي	AppliCation System
قيود عامة	Assertions
كينونة مشاركة	AssoCiative Entity
النووية	AtomiCity
خاصية (أو صفة أو سمة)	Attribute
التواجد	Availability

التصويرة السابقة	Before Image
علاقة ثنائية	Binary Relation
الشكل الطبيعي «بويس-كود»	BoyCe-Codd Normal Form ((BCNF
قواعد العمل	Business Rules
الضرب الكرتيزي	Cartesian ProduCt
منسق وسيط	CasCaded Coordinator
الفئة (أو الصنف)	Class
انغلاق	Closure
مرحلة اتخاذ القرار	Commit Phase
عبارة التثبيت	Commit Statement
نظام إدارة اتصالات	CommuniCation Manager
قواعد كاملة	Complete Rules
خاصية مركبة	Composite Attribute
النمذجة المفاهيمية	ConCeptual Modeling
نظام التحكم في التزامن	ConCurrenCy Control ProtoCol
شرط	Condition
الاسترجاع المشروط	Conditional Retrieval
الصحة (أو التوافق)	ConsistenCy
قيود الصحة (أو التوافق)	ConsistenCy Constraints
منسق	Coordinator

الاستفسارات المتداخلة المرتبطة	Correlated Nested Queries
لغة التحكم في البيانات	Data Control Language
لغة تعريف البيانات	Data Definition Language
تكرارية البيانات	Data DupliCation / Data RepliCation
هيئة البيانات	Data Format
تكامل (أو تناسق) البيانات	Data Integrity
نظام إدارة البيانات	Data Manager
لغة تداول البيانات	Data Manipulation Language
نوعية (أو نوع) بيانات	Data Type
تصميم قاعدة بيانات	Database Design
صيانة قاعدة بيانات	Database MaintenanCe
نظام إدارة قاعدة بيانات	Database Manager
لغة وصفية	DeClarative Language
قيمة افتراضية	Default Value
مشكلة الحذف	Delete Anomaly
عملية الحذف	Delete Operation
فك التطبيع	Denormalization
خاصية مشتقة	Derived Attribute

قيد الانفصال	Disjointness Constraint
نظم قواعد بيانات موزَّعة	Distributed Database Systems
معاملات موزَّعة	Distributed TransaCtions
قيود المجال	Domain Constraints
الدوام	Durability
التغليف	EnCapsulation
كينونة	Entity
نموذج «كينونة - علاقة»	Relationship Model–Entity
حدث	Event
تعبير	Expression
تنظيم الملفات	File Organization
الشكل الطبيعي الأول	(First Normal Form (1NF
تركيبة (أو صيغة)	Formula
تراكيب (أو صيغ)	Formulae
الشكل الطبيعي الرابع	(Fourth Normal Form (4NF
اعتمادية وظيفية	FunCtional DependenCy
المحافظة على الاعتماديات الوظيفية	FunCtional DependenCy Preservation
دوال	FunCtions
تعميم	Generalization

تعليمة إعطاء الصلاحية	Grant Statement
نموذج بيانات هرمي	HierarChiCal Data Model
تقسيم أفقي	Horizontal Partitioning
معرف	Identifier
خاصية مُعَرفة (أو مُمَيزة)	Identifying Attribute
ملفات مفهرسة	Indexed Files
قواعد استدلال	InferenCe Rules
مشكلة الإضافة	Insert Anomaly
عملية الإضافة	Insert Operation
تقاطع	InterseCt
عُزلة	Isolation
عملية ربط	Join Operation
مشارك أخير (أو نهائي)	Leaf PartiCipant
ربط خارجي أيسر	Left Outer Join
سجل وقائع	(Log (or Journal
تصميم منطقي لقاعدة البيانات	LogiCal Database Design
إجباري متعدد	Mandatory Many
إجباري واحد	Mandatory One
تعبير حسابي	MathematiCal Expression
تركيبة حسابية	MathematiCal Formula

عملية الفرق	Operation Minus
خاصية متعددة القيم	Multivalued Attribute
ربط طبيعي	Natural Join
استفسارات المتداخلة	Nested Queries
نموذج بيانات شبكي	Network Data Model
لغة غير إجرائية	NonproCedural Language
تطبيع	Normalization
قيود القيم الغائبة	NULL Constraints
قيمة غائبة	NULL Value
فئة (أو صنف) الشيء	ObjeCt Class
ذاتية الشيء	ObjeCt Identity
نموذج بيانات شيئي	ObjeCt-Oriented Data Model
نظم قواعد بيانات شيئية	ObjeCt-Oriented Database Systems
نظم قواعد بيانات «علاقية - شيئية»	ObjeCt-Relational Database Systems
اختياري متعدد	Optional Many
اختياري واحد	Optional One
ربط خارجي	Outer Join
انفصال متداخل	Overlapping Disjoint
اعتمادية وظيفية جزئية	Partial FunCtional DependenCy

Partial SpeCialization	تخصيص جزئي
PartiCipant	مشارك
PhysiCal Database Design	تصميم مادي لقاعدة البيانات
Polymorphism	التحميل الزائد
Prepare Phase	مرحلة التصويت
Prepare to Commit	الاستعداد للتصويت
Primary Key	مفتاح رئيسي
Privilege	صلاحية
ProCedural Language	لغة إجرائية
ProCedures	إجراءات
Program-Data DependenCe	اعتمادية (أو ترابط) بين البرامج والبيانات
ProjeCtion Operation	عملية إسقاط
Prototype	نموذج أولي (أو مبدئي أو تجريبي)
ReCovery ProtoCol	نظام استعادة (أو تشافي)
Redo Phase	مرحلة استعادة التعديلات
Referential Integrity Constraints	قيود تكامل مرجعية
Relational Algebra	الجبر العلاقي
Relational CalCulus	الحساب العلاقي
Relational Model	نموذج علاقي
Relationship	علاقة (أو ارتباط)

خصائص العلاقة	Relationship Attributes
الموثوقية	Reliability
عملية إعادة تسمية	Renaming Operation
مستودع	Repository
نظام إدارة موارد	ResourCe Manager
تعليمة سحب صلاحية	Revoke Statement
ربط خارجي أيمن	Right Outer Join
إجراءات متكررة	Routines
تعبير آمن	Safe Expression
الشكل الطبيعي الثاني	(SeCond Normal Form (2N
مفتاح ثانوي	SeCondary Key
تعليمة اختيار (أو استرجاع أو استفسار)	SeleCt Statement
ملف متسلسل	Sequential File
خاصية بسيطة	Simple Attribute
قواعد سليمة	Rules Sound
تخصيص	SpeCialization
قيد تخصيص	SpeCialization Constraint
سجلات زائفة	Spurious Tuples
كينونة قوية	Strong Entity
لغة استفسار بنائية	(StruCtured Query Language (SQL

نوع فرعي	Subtype
نوع رئيسي	Supertype
مفتاح خارق	Supper Key
مطوِّر نظم	System Developer
دورة حياة تطوير نظام	System development Life CyCle ((SDLC
علاقة ثلاثية	Ternary Relation
الشكل الطبيعي الثالث	(Third Normal Form (3NF
المنطق الثلاثي القيم	Three-Valued LogiC
انفصال كامل	Total Disjoint
تخصيص كامل	Total SpeCialization
معاملة	TransaCtion
نموذج تنفيذ معاملات	TransaCtion ExeCution Model
نظام إدارة معاملات	TransaCtion Manager
اعتمادية وظيفية انتقالية	Transitive FunCtional DependenCy
زنادات	Triggers
قيود سجلات	Tuple Constraints
متغيرات سجلات	Tuple Variables
بروتوكول تثبيت ذو مرحلتين	Two-Phase Commit ProtoCol
علاقة أحادية	Unary Relation
مرحلة إلغاء التعديلات	Undo Phase

اتحاد	Union
مشكلة التحديث	Update Anomaly
تعليمة التحديث (أو التعديل)	Update Statement
تقسيم رأسي	VertiCal Partitioning
منظور	View
كينونة ضعيفة	Weak Entity
تدوين مُسبق للوقائع	Write-Ahead Logging

المراجع:

- Agrawal, S., V. Narasayya and B. Yang. “Integrating VertiCal and Horizontal Partitioning into Automated PhysiCal Database Design,” ProCeedings of the 2004 ACM SIGMOD International ConferenCe on Management of Data, Paris, FranCe, June 2004.

- Al-Houmaily, Yousef J. “AtomiC Commit ProtoCols, their Integration, and their Optimisations in Distributed Database Systems,” International Journal of Intelligent Information and Database Systems, Vol. 4, No. 4, 2010.

- Al-Houmaily, Yousef J. “On Deferred Constraints in Distributed International Journal of Database Management”Database Systems, Systems, Vol. 5, No. 6, DeCember 2013.

- Al-Houmaily, Yousef J. “GLAP: A Global LoopbaCk Anomaly ”Prevention MeChanism for Multi-Level Distributed TransaCtion, International Journal of Database Management Systems, Vol. 6, No. 3, June 2014.

- Al-Houmaily, Yousef J. and George Samaras. “Two-Phase Commit,” In EnCyClopedia of Database Systems, Ling Liu and Tamer M.

Özsu (Eds.), 2nd edition, 2018.

- Bernstein, P. A., V. HadzilaCos and N. Goodman. “ConCurrenCy Addison-Wesley, 1987.”Control and ReCovery in Database Systems,

- BruCe, T. A. “Designing Quality Databases with IDEFIX Information Models,” New York: Dorset House, 1992.

- Cannan, Stephen and Gerard Otten. “SQL - The Standard Handbook,” MCGraw-Hill Book Company, 1993.

- Codd, E. F. “A Relational Model of Data for Large Relational Databases,” CommuniCations of the ACM, Vol. 13, June 1970, pp. 377-397.

- Connolly, Thomas and Carolyn Begg. “Database Systems: A 6th”PraCtiCal ApproaCh to Design, Implementation, and Management, Edition, Essex, England: Pearson EduCation Limited, 2015.

- Efraim, Turban, Ephraim MClean and James Wetherbe. 3rd Edition, New York: John”“Information TeChnology for Management, Wiley & Sons, InC., 2002.

- Elmasri, Ramez and Shamkant Navathe. “Fundamentals of Database Systems,” 7th Edition, Pearson EduCation, InC., 2015.

- Fleming, C. and B. von Halle. “Handbook of Relational Database Design,” Reading, MA: Addison-Wesley, 1989.

- Frank, M. “ObjeCt-Relational Hybrids,” DBMS, July 1995, pp. 46-56.

- GarCia-Molina, HeCtor, Jeffery Ullman and Jennifer Widom.
Harlow: Pearson EduCation""Database Systems: The Complete Book,
Limited, 2014.

- Gray, Jim and Andreas Reuter. "TransaCtion ProCessing: ConCepts
Morgan Kaufmann, San Mateo, CA, 1992."and TeChniques,

- Hoffer, Jeff, Ramesh Venkataraman and Heikki Topi. "Modern
13th Edition, Pearson, 2018."Database Management,

- Inmon W. H. "What PriCe Normalization," Computer World, Vol.
27, No. 31, 1988.

- Martin, James. "Information Engineering: Book I, IntroduCtion,"
New Jersey: PrentiCe-Hall, InC., 1989.

- Ramakrishnan, Raghu and Johannes Gehrke. "Database
Management Systems," 3rd Edition, MCGraw-Hill Higher EduCation,
2003.

- Rogers, u. "Denormalization: Why, What and How?" Database
Programming and Design 2, pp. 46-53, DeCember 1989.

- Stallings, William. "LoCal and Metropolitan Area Networks," 4th
Edition, MaCmillan Publishing Company, 1993.

المؤلف في سطور

أ.د. يوسف بن جاسم بن محمد الهميلي.

المؤهل العلمي:

- دكتوراه في هندسة الحاسب الآلي من جامعة بتسبرغ، عام 1997م.

الوظيفة الحالية:

- أستاذ هندسة الحاسب الآلي ونظم المعلومات بمعهد الإدارة العامة بمدينة الرياض.

أبرز الخبرات العلمية والعملية:

- يحتوي سجله البحثي حالياً على أكثر من عشرين بحثاً ومقالةً علميةً منشورة بالإضافة إلى كتابين. ويُعَدُّ بعضُها مرجعاً في تخصُّص نظم قواعد البيانات على المستوى الدولي. كما أنه عضو في هيئتَيْ تحرير مجلتين علميتين في مجال تخصُّصه، وعضو في اللجان المنظمة لعددٍ من المؤتمرات الدولية. إضافةً إلى ذلك؛ فهو عضو في بعض الجمعيات المهنية.

- مكَّنه سجله البحثي من المشاركة في كتابة فصول في كتب علمية متخصِّصة، ومن المشاركة أيضاً في تأليف موسوعة نظم قواعد البيانات (EnCyClopedia of Database Systems) بطبعتيها الأولى (2009م)، والثانية (2018م).

- تقع اهتماماته البحثية حالياً في مجال نظم قواعد البيانات، ونظم الحوسبة المتنقلة الموزَّعة، وشبكات المجسَّات.

- عمل مديراً لبرامج الحاسب الآلي والمعلومات بمعهد الإدارة العامة مدة خمس سنوات (في الفترة من 1997م وحتى 2002م)، وهو عضوٌ في عددٍ من اللجان والمجالس في المعهد؛ حيث عمل ضِمن المجلس العلمي (لست دورات)، واللجنة الدائمة للترقيات العلمية المنبثقة عن المجلس لفترةٍ مماثلة. كما ترأس فريقَ إعداد الخطة الإستراتيجية الأولى للمعلومات والتعاملات الإلكترونية للمعهد.

- عمِل على المستوى الوطني في عددٍ من اللجان وفِرق العمل؛ من ضِمنها: عضويته في فريق دراسة وزارة الداخلية والمناطق الإدارية المشكَّل من قِبل الأمانة العامة في اللجنة الوزارية للتنظيم الإداري، ولجنة تقييم عَرْض شركة ميكروسوفت العالمية لحكومة المملكة العربية السعودية؛ لتعميم استخدام منتجاتها البرمجية على الأجهزة الحكومية في المملكة.

- عمِل مستشاراً متفرغاً في ديوان سموّ ولي العهد، ومستشاراً غير متفرغ في هيئة الرقابة والتحقيق، ومستشاراً غير متفرغ في الرئاسة العامة لتعليم البنات.

- قام بتدريس مواد متخصِّصة في الحاسب الآلي داخل المملكة العربية السعودية، والولايات المتحدة الأمريكية، وكندا. كما عمِل أستاذاً زائراً في جامعة واترلوو الكندية؛ حيث قام بتدريس مواد متخصِّصة في نظم قواعد البيانات على مستوى الدراسات الجامعية (البكالوريوس) وعلى مستوى الدراسات العليا.

تم التصميم والإخراج الفني والطباعة في
الإدارة العامة للطباعة والنشر - معهد الإدارة العامة 1442هـ

Notes

[1←]

تُستخدَم كلمة «FaCulty» لتعني «كلية» أو كافة التابعين لها من طلبة وموظفين على اختلاف طبيعة أعمالهم، إلا أن هذه الكلمة تُستخدَم أيضاً في شمال أمريكا (الولايات المتحدة الأمريكية وكندا) للدلالة على الموظفين في حقل التعليم وخاصةً الجامعات والكليات العلمية الذين يقومون بالتدريس دون سواهم من الموظفين الذين لا يقومون بمهامِ التدريس. وفي هذه الحالة تكون الكلمة مكافئةً لكلمة «أستاذ» (Professor) أو محاضر (LeCturer).

[2←]

قد لا يكون لمدير المنظمة مَنْ يرأسه وفي هذه الحالة يتمُّ إدخال قيمة مساوية لرقم الموظف في حقل المفتاح الخارجي؛ للدلالة على أن الموظف يدير نفسه مع ضرورة تعطيل العمل في القيود في أثناء إدخال سجل لمثل هذا الموظف، كما سنُوضِّح في الفصل السابع عند شرح طريقة تعطيل العمل بالقيود. وكبديلٍ لذلك يُمكِن تعريف العلاقة بأنها اختيارية عوضاً عن كونها إجبارية (واحد).